Alejandro Anton Turc

Grau en Enginyeria en Tecnologies Industrials

Juliol 2015

Treball de Fi de Grau

Grau en Enginyeria en Tecnologies Industrials

Desenvolupament d'aplicacions per mòbil amb Kivy

MEMÒRIA

Autor: Alejandro Anton Turc
Director: Lluis Solano Albajes

Convocatòria: Juliol 2015

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Resum

Aquest projecte de fi de grau consisteix en l'elaboració del prototip d'un joc multi-jugador en temps real per mòbil completament desenvolupat amb el llenguatge de programació Python. També consisteix en la creació d'un servidor per ordinador que s'encarrega d'administrar les connexions i les partides que es donen a terme dins el joc entre els diferents jugadors.

Una part molt important d'aquest treball és mostrar la capacitat que té Python com a llenguatge de programació així com el potencial de les llibreries que s'han fet servir, ja que tot i no ser el llenguatge més òptim en alguns camps, pot arribar fins al punt que es vulgui.

El resultat del projecte ha estat un joc per a dispositius Android que permet als usuaris fer petites batalles entre ells amb naus espacials. Els usuaris estaran en un planeta competint amb altres jugadors fins que guanyin o perdin i tindran algunes opcions a triar.

Per tal de dur a terme aquesta aplicació s'han estudiat i usat diferents llibreries de Python. S'ha usat la llibreria *Kivy* per a l'aplicació mòbil, la llibreria *PyQt4* per a la interfície gràfica del servidor multi-jugador, i el mòdul *Twisted* per a la creació i administració de les connexions web entre aplicació mòbil i servidor. També s'ha utilitzat la llibreria *SQLAlchemy* per a les qüestions de la base de dades MySQL on es guarden dades del joc.

En aquest treball es comenten aquestes llibreries i a l'annex s'explica detalladament com es poden fer servir i com funcionen. En aquest treball s'explica també com s'ha fet l'aplicació mòbil i l'aplicació servidor i com funcionen conjuntament per permetre el joc multi-jugador. També s'explicarà com es pot disposar de l'aplicació feta amb Python al mòbil.

Sumari

1. Glossari

Al llarg del treball es poden trobar alguns termes o expressions típics de la programació i de la informàtica així com del propi llenguatge Python. A continuació s'expliquen alguns d'ells:

- TCP: Protocol de comunicació web que es basa en la creació d'una connexió entre dos dispositius, garantint l'arribada de les dades que s'envien i en el mateix ordre. Si s'envia una dada no es perdrà i arribarà sempre a menys que es perdi la connexió.

- UDP: Protocol de comunicació web que es basa en l'enviament de dades d'un dispositiu a un altre sense haver creat cap connexió. Aquestes dades poden arribar en un ordre diferent al què s'han enviat i poden perdre's pel camí. Tot i això la comunicació és més ràpida.

- IP estàtica i dinàmica: Tipus de IP's que existeixen. Es pot tenir una IP estàtica, la qual serà sempre la mateixa, o una IP dinàmica, que varia sola segons certes necessitats.

- Multi-touch: Característica d'una aplicació en la què es poden realitzar varies accions de tocar la pantalla de manera simultània i poden ser processades a la vegada.

- Herència: Característica de les classes de Python en què una classe comparteix els mètodes i atributs d'una altre de la qual hereta.

- APK (Android application Package): És el tipus de format dels arxius que serveixen per instal·lar i distribuir les aplicacions en el sistema operatiu d'Android.

- FPS (Frames per second): Nombre de frames per segon al què treballa una operació. Significa el nombre de cops per segon que s'actualitzen els paràmetres d'alguna funció.

- Bucle infinit de control: Expressió que denomina el flux de control que té una aplicació. Algunes aplicacions i programes informàtics s'executen un cop i després s'acaben, però d'altres estan funcionant contínuament i és el propi programa el que s'encarrega d'anar fent "voltes" actualitzant i processant totes les accions i esdeveniments que hi tenen lloc.

- Fitxer ".py": Format que denomina els fitxers de text on hi ha escrit codi Python i què pot ser executat posteriorment pel compilador.

- Fitxer ".ui": Format que denomina els fitxers resultants del programa Qt Designer per la creació d'interfícies gràfiques. Posteriorment seran convertits a fitxers ".py".

- Primary Key: Característica d'una columna d'una taula d'una base de dades SQL que indica que aquella columna és la principal i que es farà servir per accedir a cada fila.

2. Prefaci

2.1. Origen i motivació del projecte

La motivació principal per a la realització d'aquest projecte ha estat el meu afecte per a la programació i en especial amb el llenguatge Python. Vaig començar a programar amb les assignatures de primer i segon any del grau d'Enginyeria en Tecnologies Industrials i em va atraure força el què fèiem. El meu nivell de programació va créixer exponencialment quan vaig acabar segon, al haver cursat en el mateix quadrimestre l'assignatura optativa de **Jocs per computadors** (Llenguatge *Actionscript*) i **Disseny i desenvolupament de software amb Python** a Projecte I. A les dues assignatures s'havia de crear una aplicació i no teníem límits per fer-les, i això es el que més em va agradar.

Hi ha altres àmbits de la enginyeria que m'atrauen molt, i per això no vaig escollir el grau d'informàtica, però la llibertat que dona aquest camp per expressar i crear el que es vulgui és el que m'atrau més. No hi ha cap altre camp on es puguin expressar les idees i creacions que sorgeixen de manera més fàcil que amb la programació.

Des d'abans de començar aquest projecte que havia estat investigant el mòdul per aplicacions per mòbil *Kivy*, i havia fet només una petita aplicació que es pot trobar al Google Play Apps. Com ja estava breument introduït en aquest camp volia investigar més a fons i volia veure si podia relacionar-ho amb altres mòduls de la informàtica que també m'interessaven i que tenen els seus mòduls i llibreries a Python, com les bases de dades, els programes amb interfície gràfica, o les connexions entre dispositius per internet.

3. Introducció

El món de les aplicacions per mòbil és un sector molt important avui en dia. La major part de la població disposa de telèfons intel·ligents i utilitza un gran ventall d'aplicacions diferents a diari, i entre elles, abunden els jocs. La oferta de jocs és una de les més àmplies ja que actualment els videojocs són el negoci que mou més diners del món. Es creen jocs de totes les categories i gustos i per a tots els públics, i això fa que sigui molt atractiu per als desenvolupadors d'idear nous jocs dia a dia.

Hi ha diferents llenguatges de programació per desenvolupar aplicacions per mòbil, els més comuns Java d'Android i Objective C de IOS. Tot i això, per què no desenvolupar aplicacions amb el llenguatge més popular avui en dia, el Python? Al tractar-se d'un llenguatge molt utilitzat sempre es poden trobar biblioteques per tots els camps que altre gent ja ha creat i que posa a la disposició del públic, i és molt útil.

En aquest projecte es realitza una aplicació per Android completament en Python amb la llibreria Kivy i es barreja amb altres llibreries d'altres camps de la informàtica. Avui en dia hi ha moltes aplicacions que disposen d'accés a internet i que permeten guardar informació i dades de l'aplicació que s'han modificant per l'usuari. Aquesta informació s'acostuma a guardar a una base de dades que el proveïdor de l'aplicació té funcionant.

Tots aquests camps són molt interessants i estaria molt be aconseguir desenvolupar una aplicació amb aquest llenguatge de programació que els tingués presents.

3.1. Objectius del projecte

L'objectiu principal del projecte és la creació i desenvolupament del prototip d'un joc multi-jugador en temps real per Android. L'aplicació es desenvoluparia amb el llenguatge de programació Python i permetria als usuaris jugar partides online mitjançant un servidor creat també amb aquest llenguatge que administraria les connexions i partides.

En segon lloc, es vol obtenir l'estudi i la valoració del fet de crear aquest software amb un llenguatge de programació que no és l'habitual per aquest tipus d'aplicacions.

Com un dels punts més importants, es comentaran i valoraran les llibreries utilitzades per al desenvolupament d'aquest software i s'intentarà facilitar el treball fet amb aquestes llibreries de manera de puguin ser fàcilment reaprofitades en un futur per qualsevol persona que vulgui desenvolupar aplicacions amb qualsevol d'elles. S'explicaran els punts forts i interessants de totes elles perquè puguin ser utilitzats i entesos fàcil i ràpidament.

Dos objectius de cara a la creació del software són la introducció a la comunicació web entre dispositius i la introducció a les bases de dades MySQL, tot relacionant-ho amb el joc per mòbil. Es vol aconseguir plantejar i portar a terme una arquitectura de comunicacions que aconsegueixi establir una connexió entre diferents dispositius per enviar dades. Es voldrà aconseguir emmagatzemar des de el mateix codi de programació les dades que interessin a la base de dades MySQL. Aquesta arquitectura es provarà amb el joc creat.

Amb el software finalitzat també està present l'objectiu de transformar l'aplicació feta a Android per poder penjar-la al web i fer-la accessible al públic.

3.2. Abast del projecte

L'abast del projecte arriba fins a la creació del prototip del joc multi-jugador així com fins a la creació del servidor. El joc permetrà als usuaris registrar-se i efectuar partides multi-jugador que controlarà el servidor i estarà disponible a la web mitjançant la plataforma Google Play. El joc disposarà d'un mode de joc senzill i en proves, amb bastantes qüestions desenvolupades que es comentaran més tard. A l'annex es podrà trobar un estudi detallat com utilitzar les diferents llibreries que s'han fet servir pel desenvolupament del joc i de cara a la defensa del treball es farà una demostració d'una partida multi-jugador i es mostraran les accions que hi tenen lloc, tant al joc com al servidor.

3.3. Antecedents

La majoria de les aplicacions avui en dia es desenvolupen amb els llenguatges de programació *Java* (per Android) i *Objective C* (per IOS) tot i que avui en dia també és molt important UNITY, una plataforma que treballa amb *Javascript* i *C#* i que permet crear aplicacions que amb certes modificacions poden ser distribuïdes a quasi totes les plataformes que hi ha al mercat, incloent tant Android com IOS i altres com Windows o Play Station. També hi ha altres plataformes de desenvolupament de jocs com *Cocos3D* o *Kivy.*

Kivy és una llibreria per desenvolupar aplicacions per mòbil completament en Python i cada dia és més utilitzada i disposa de millor documentació. Tot i això té certes qüestions que s'analitzaran i es comentaran que podrien ser millorades com per exemple la transformació i conversió dels programes desenvolupats cap a Android. És una eina potent que juntament amb el potencial del Python obre moltes portes als desenvolupadors.

4. Descripció de l'arquitectura de l'aplicació

Al principi del projecte es va començar a pensar com es podia dur a terme el desenvolupament de la idea d'un joc multi-jugador per mòbil amb Python. Inicialment només es pretenia fer un joc en què els jugadors tinguessin les seves dades i característiques de joc guardades en una base de dades a la què s'accediria directament des de l'aplicació. Això va resultar inviable ja que no és recomanable que una aplicació mòbil accedeixi directament a una base de dades. Més endavant, i amb la necessitat d'un servidor que fes d'intermediari entre mòbil i base de dades, es va decidir que el joc es podria fer multi-jugador en temps real. L'estructura inicial que es va pensar va ser de fer un servidor de Python a l'ordinador amb accés a una base de dades local que es comuniqués amb els mòbils. A la figura 4.1 es pot veure aquest esbós de l'estructura inicial.

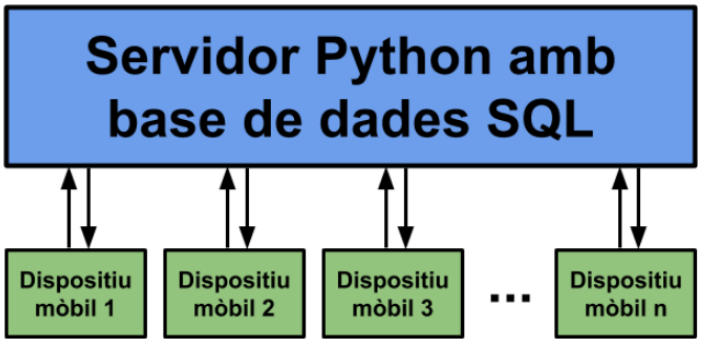

Fig. **4.1.** Esquema de l'esbós de l'estructura de inicial l'aplicació conjunta servidor-mòbils

Per aconseguir portar a terme el concepte ideat s'havia de buscar la manera i les eines per fer-ho, i com ja estava decidit que es faria amb Python, es va fer una exhaustiva recerca de llibreries i biblioteques per veure què s'oferia sobre accessos a bases de dades i sobre comunicació web. Per a l'aplicació mòbil estava clar que es faria servir *Kivy,* ja coneguda, i pels accessos de bases de dades es van trobar diverses biblioteques com *PyMySQL, SQLAlchemy, MySQLdb o MySQL-python.* Després d'analitzar-les es va triar utilitzar *SQLAlchemy* ja que és la llibreria més complementada i ben valorada així com la llibreria de més alt nivell a nivell d'estructures de Python a utilitzar. Per a la comunicació web es van trobar dos llibreries principals a analitzar, *Twisted* i *Flask* , ja que de bon principi ja s'havia descartat la llibreria *Django* que treballa bàsicament amb servidors de pàgines web i s'escapava en quan a continguts i complexitat. De les dos opcions es va acabar escollint *Twisted* degut a què els serveis que ofereix s'adeqüen més a la idea d'aconseguir una connexió directa entre diferents dispositius. *Flask* per altre banda, es va veure que és una llibreria molt potent però que està més basada en crear petits servidors web al que els usuaris poden fer-li peticions (*requests*) per accedir o modificar dades.

Així doncs, com es pot veure a la figura 4.2, aquesta és l'estructura que ha acabat tenint l'aplicació amb les llibreries escollides a cada part, així com les connexions existents. Per a la comunicació mòbil-servidor s'ha escollit el protocol de comunicació TCP i el port 7788. S'ha creat un domini que fa de pont cap al servidor establert a l'ordinador de casa evitant els possibles canvis d'IP dinàmica. Per definir el domini gratuït *alejandroserver.sytes.net* s'ha utilitzat el programa NoIP DUC que s'explicarà en el següent capítol.

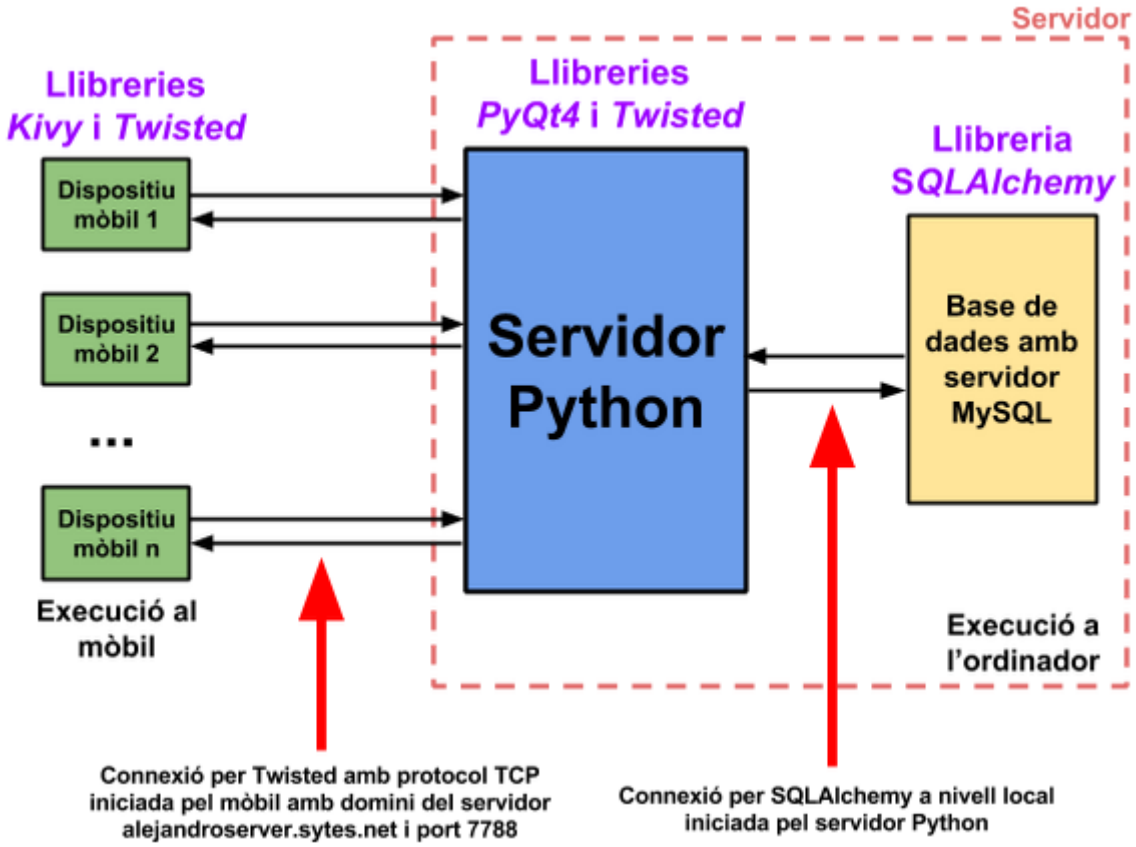

Fig. **4.2.** Esquema final de l'estructura de l'aplicació amb les seves connexions i llibreries

Amb aquesta estructura s'aconsegueix que en el moment que es realitza una partida entre dos jugadors el que fan és enviar-se constantment dades entre ells mitjançant el servidor per aconseguir compartir l'experiència en temps real de la mateixa partida.

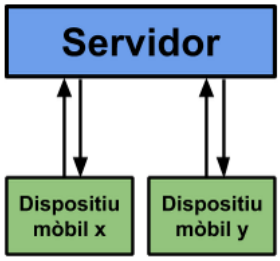

Fig. **4.3.** Esquema de l'estructura de les connexions en una partida amb dos jugadors

5. Eines

5.1. Llenguatge i sistema operatiu

Aquest projecte s'ha dut a terme completament en el llenguatge de programació Python. Python és un llenguatge de programació de codi obert molt conegut i utilitzat i és el més popular avui en dia. És un llenguatge que destaca per la clara sintaxi i compressió que presenta així com per la seva senzillesa a la hora de desenvolupar aplicacions i de manipular d'elements.

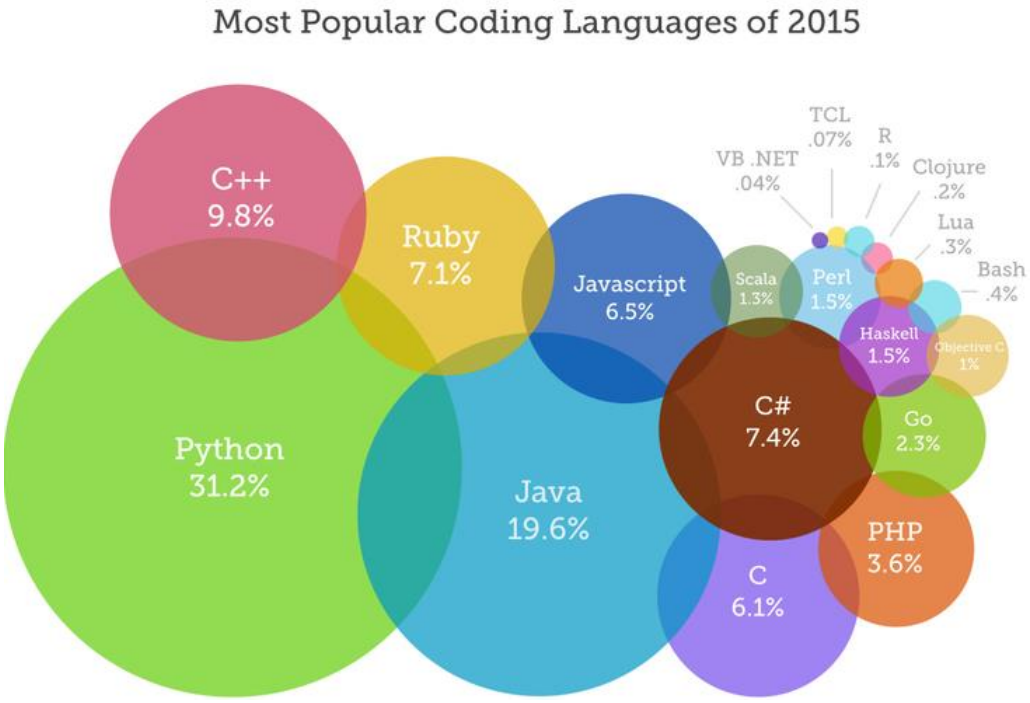

Fig. **5.1.** Digrama que mostra la popularitat dels llenguatges de programació al 2015. **[1]**

En tota la realització del treball s'ha treballat amb el sistema operatiu Windows. Tot i no estar qualificat com la millor alternativa per a treballar amb Python, ja que per exemple amb Linux podria semblar més fàcil, s'ha decidit treballar amb Windows ja que és el que sempre m'ha semblat més còmode i amb el que tinc més experiència i domini.

Tot el codi que s'ha programat pel desenvolupament d'aquest treball s'ha creat de la manera més òptima possible. No només s'ha optimitzat el codi de manera que sigui agradable a nivell visible sinó que el codi en si està optimitzat a nivell d'evitar el màxim redundàncies i excés de treball dels dispositius realitzant càlculs.

5.2. Programes utilitzats

S'han utilitzat alguns programes i aplicacions pel desenvolupament de diferents parts del treball per complir amb el procés ideat i s'expliquen i es mostren a continuació:

- **Qt Designer:** Programa que permet la creació de l'estructura de les interfícies gràfiques amb què treballa la llibreria PyQt4. Aquest programa permet estructurar d'una manera molt visual es vol la situació dels elements de les finestres que es creen amb Qt. Un cop estructurat es convertirà el fitxer resultant d'aquest programa a codi Python per ser inclòs com un mòdul més a la llista de fitxers ".py". S'ha fet servir per dissenyar l'estructura visual del servidor multi-jugador.

- **Kivy Launcher:** Aplicació per mòbil creada per Kivy que permet visualitzar al dispositiu l'aplicació creada amb Python sense necessitat d'haver-la convertit a Android. L'únic que fa falta és passar els fitxers ".py" al mòbil amb per USB i obrir el Kivy Launcher, tot seleccionant quina aplicació es vol executar, si és que hi ha més d'una. S'ha fet servir per fer proves amb les aplicacions en diferents dispositius.

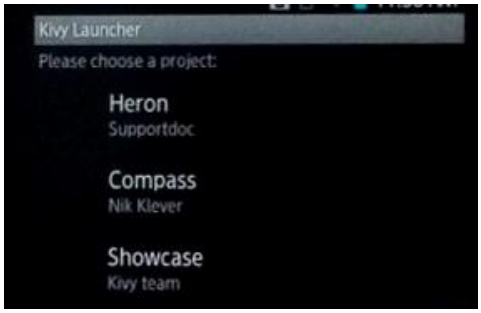

Fig. **5.2.** Imatge de mostra de l'aplicació Kivy Launcher per triar l'aplicació a emular

- **NoIP DUC:** Programa que permet treballar amb tots els serveis que ofereix la plataforma NoIP DUC. Aquests serveis serveixen per treballar amb dominis i IPS i s'ha fet servir per crear un domini del servidor evitant els canvis de la IP dinàmica. Això és molt útil perquè encara que variïn les IP's, les aplicacions tindran clar on connectar, allà on aquest programa estigui encès amb el domini corresponent.

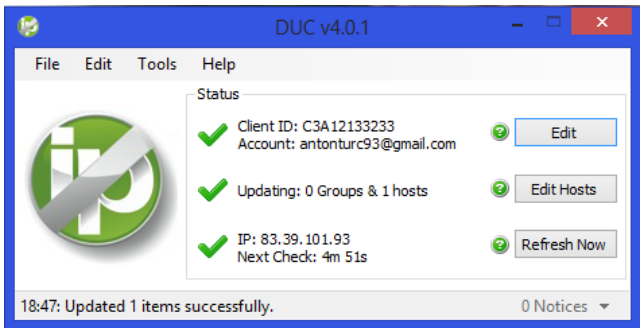

Fig. **5.3.** Captura de pantalla del programa NoIP DUC

- **Oracle VM VirtualBox:** Programa d'Oracle que permet treballar amb discs virtuals permetent utilitzar qualsevol sistema operatiu com si fos una simple finestra. Aquests discs virtuals s'han de crear o descarregar, i a continuació configurar, i acostumen a tenir una mida elevada. S'ha utilitzat per treballar amb un disc virtual d'Ubuntu preparat per a la conversió de Python a Android.

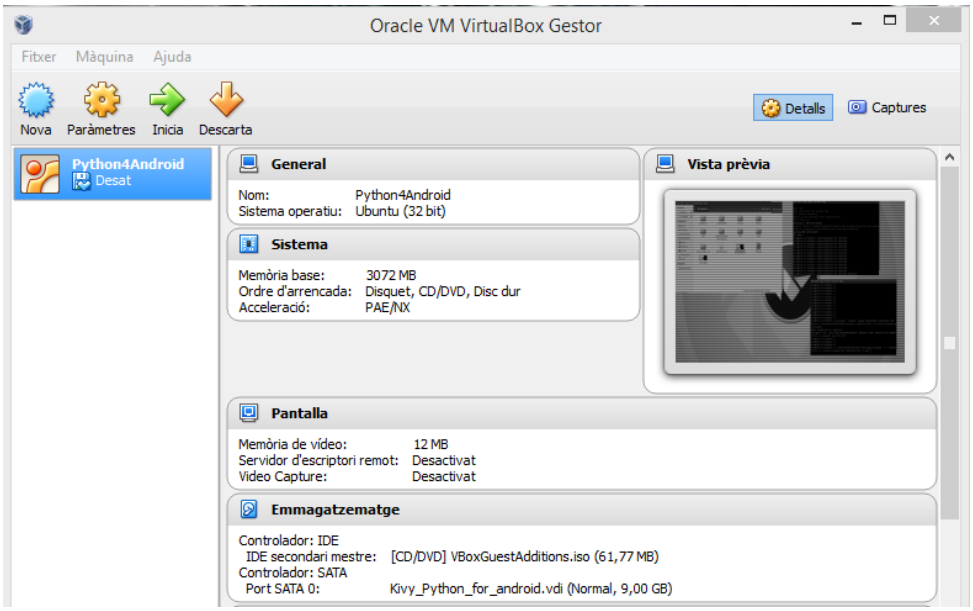

Fig. **5.4.** Captura de pantalla del programa Oracle VM VirtualBox abans de triar disc

- **MySQL WorkBench:** Programa que permet treballar a nivell manual amb les bases de dades MySQL. Aquest programa no només serveix per crear el servidor MySQL al qual s'accedirà, si no que també permet veure, accedir i modificar les bases de dades amb les corresponents taules i columnes.

- **Eclipse:** Programa d'edició de text dissenyat principalment per treballar amb Java. Amb unes llibreries que es poden instal·lar es pot configurar per treballar amb Python. S'ha utilitzat com a programa d'edició de text i compilació dels fitxers ".py".

5.3. Altres

S'ha treballat amb un ordinador de taula amb un processador i7 de 3.4 GHz, amb 12Gb de memòria RAM, amb dos discs durs sòlids de 256Gb cadascun, i amb una targeta gràfica NVIDIA GeForce (GTX 650) de 2Gb. S'han fet servir també un dispositiu mòbil i una tableta per a les proves de l'aplicació, ambdós Samsung amb sistema operatiu Android, utilitzant la tableta només per a les proves de competició en el mode multi-jugador.

6. Llibreries

En aquest apartat es mostren totes les llibreries de Python que s'han utilitzat pel desenvolupament de l'aplicació mòbil i del servidor així com d'altres llibreries auxiliars. Aquí es comentaran i a l'annex es poden trobar explicacions detallades del seu funcionament.

6.1. Kivy

Kivy és una llibreria multi-plataforma que permet crear aplicacions amb interfície gràfica. Permet crear aplicacions *multi-touch* per a diferents plataformes com IOS o Android i està definit com un projecte de comunitat portat per desenvolupadors de software professionals. És la llibreria base en què es basa l'aplicació mòbil. La primera versió va sortir al 2011 i avui en dia disposa d'àmplies biblioteques i documentació que expliquen amb molt detall què es pot fer i de quina manera. És una llibreria de codi obert que ofereix tot un seguit de classes molt complertes i ben documentades per tal de desenvolupar amb una gran llibertat aplicacions completament amb Python.

Kivy té una manera de funcionar molt semblant a la d'altres llibreries de construcció d'interfícies gràfiques com per exemple *PyQt4* en el sentit que tot funciona a base de jerarquies d'elements. Tots els elements i classes que es trobaran hereten d'altres així que sempre tenen mètodes i atributs en comú que poden ser modificats en qualsevol moment en qualsevol punt de l'aplicació. Algunes de les classes que hi ha s'acostumaran a instanciar directament, però en canvi amb la majoria interessarà crear les nostres pròpies classes heretant de les de *Kivy* per tenir tots els atributs i mètodes que es vulguin ja creats.

Tot i tenir certes qüestions i apartats que deixen una mica que desitjar i que podrien fer-se d'una manera més fàcil, quan es tracta de les qüestions essencials i d'aplicacions senzilles hi ha prou amb la senzillesa de Python per aconseguir resultats.

```
from kivy.app import App
from kivy.uix.button import Button

class TestApp(App):
    def build(self):
        return Button(text="Hello World")

TestApp().run()
```

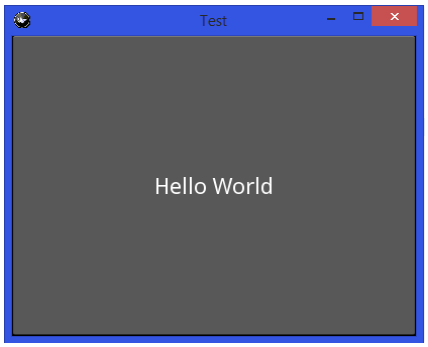

Fig. **6.1.** Exemple del codi i del resultat d'una simple aplicació amb un botó

6.2. PyQt4

PyQt4 és una llibreria de creació d'interfícies gràfiques traduïda del QtProject de C++. Disposa d'una gran quantitat d'elements i classes que permeten utilitzar quasi qualsevol objecte que es podria trobar en una aplicació, com finestres de diversos tipus, botons, espais de text, etiquetes, etc... La primera versió de la llibreria va sortir al 1999 i cada cop està més en desús. Es una biblioteca que té poques actualitzacions i revisió tècnica perquè hi ha altres alternatives per al desenvolupament d'aplicacions amb interfícies gràfiques com altres llenguatges de programació per fer aquestes tasques que la gent troba millors.

Es tracta d'una llibreria que funciona a partir de jerarquies d'elements en el sentit que tota classe hereta d'altres per compartir elements i mètodes. Com amb *Kivy* això és molt útil per compartir mètodes i atributs.

El treball a fer amb aquesta llibreria es podria separar en dos parts. Una part consisteix en la creació de la pròpia interfície amb els elements propis de *PyQt4*. Aquesta part es pot fer a mà cridant els diferents elements un per un o es pot fer utilitzant el programa Qt Designer que permet dissenyar l'estructura que es vol que tinguin les finestres. L'altre part de la feina a fer seria desenvolupar què vol que faci l'aplicació. Aquesta tasca consisteix en definir les interaccions dels usuaris amb diferents elements de la interfície com botons o altres elements i lligar-los amb altres funcions de Python que contenen el codi de les tasques a desenvolupar. Aquesta part inclou també per tant la creació i definició de les funcions que contindran el cos del programa.

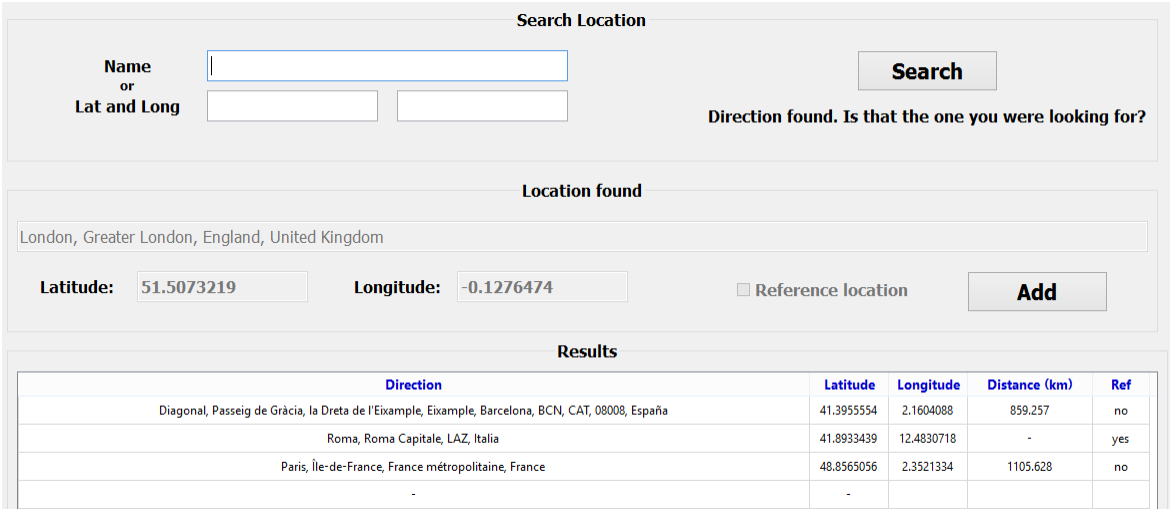

Fig. **6.2.** Interfície gràfica desenvolupada amb PyQt4 per a la optativa de Logística

6.3. Twisted

Twisted és una llibreria que representa un motor de creació i administració de xarxes i connexions orientat a esdeveniments escrit en Python. La primera versió va sortir al 2002 i és de codi obert.

Es tracta d'una llibreria molt potent per a la creació de connexions entre dispositius per aconseguir l'enviament de dades. Disposa d'una ample llista de classes i funcions que permeten utilitzar molts tipus de protocols (TCP, UDP, SSL, etc...) i diferents maneres i formes de treballar amb la xarxa. Conté una gran varietat d'elements per administrar clients i servidors, per treballar amb protocols d'e-mail o missatgeria instantània, i per treballar amb DNS.

Tot i disposar d'una extensa documentació és més difícil aprendre i dominar el seu funcionament que altres llibreries ja que requereix estar introduït en el coneixement de tecnologies de xarxes i comunicacions web. També requereix més dedicació ja que les classes i funcions de què disposa, així com la seva documentació, s'han d'analitzar molt a fons en quan a contingut i ús. Fins hi tot hi ha funcions que els propis desenvolupadors de *Twisted* recomanen no analitzar a fons per no trencar-se massa el cap.

Aquesta llibreria d'alt nivell es basa bàsicament en la recopilació i creació de software fent servir les llibreries *socket* i *thread*, que són més de nivell baix i són pròpies de Python. Es podria utilitzar perfectament comunicació web amb només aquests dos mòduls però *Twisted* facilita molt la feina i ho dona tot fet ja.

Twisted permet crear programes que es poden definir com a servidors o com a clients. Disposa de codi que permet que una aplicació estigui escoltant per a l'arribada de connexions entrants o permet que l'aplicació intenti connectar-se amb algú que estigui escoltant. Disposa de moltes funcions i mètodes per analitzar i tractar les comunicacions de diferents maneres i disposa de moltes funcions que es cridaran de manera automàtica quan tinguin lloc certes eventualitats com la creació o la pèrdua d'una connexió, l'arribada d'una dada, un error en la connexió existent, etc...

6.4. SQLAlchemy

SQLAlchemy és una llibreria d'alt nivell que serveix per a la creació, anàlisi i administració de bases de dades SQL. La primera versió va sortir al 2006.

És una llibreria molt potent i avançada que permet treballar amb diferents dialectes SQL com són per exemple MySQL, Sqlite, PostgresSQL o Oracle. Es basa en la llibreria de baix nivell *MySQLdb* que permet també el control de les bases de dades però d'una manera més simple i manual, executant comandes (*querys*) de SQL amb text pur i dur. *SQLAlchemy* en canvi, treballa amb classes i funcions que permeten fer d'una manera més automatitzada i senzilla els accessos que es voldran fer a la base de dades.

Cal remarcar que en el cas de treballar amb MySQL ,*SQLAlchemy* **no** crea un servidor SQL (i en el cas de PostgresSQL potser tampoc). L'únic que fa és accedir al servidor en cas que existeixi i pugui connectar-se, i per crear-lo, fan falta programes com MySQL WorkBench o Xampp.

Les bases de dades en format SQL es divideixen en elements, els més importants són les bases de dades, les taules, i les columnes. Poden haver-hi diferents bases de dades, cadascuna d'elles amb diferents taules, i cada taula amb diferents columnes. Cada cop que s'afegeix una fila a una taula d'una base, significa que s'està afegint una dada que tindrà tants elements com columnes tingui la taula.

	accounts	games
	5	2
▶	6	1

Fig. **6.3.** Exemple d'una petita taula SQL amb les columnes *accounts* i *games* i dues files

6.5. Interaccions entre llibreries

Hi ha llibreries utilitzades que s'han hagut d'unir i que s'han hagut de fer interactuar i això no és sempre senzill d'aconseguir i acostuma a portar mal de caps.

Per una banda, si s'han d'unir dos llibreries diferents i conegudes que ja estan incorporades a Python com per exemple *Math* i *Numpy*, és pot fer quasi bé sense pensar. El que s'ha de fer es importar els dos mòduls i fer-los servir com es vulgui (i instal·lar-los prèviament si és necessari com *Numpy*).

Per altre banda, si es tracta de llibreries complexes que tenen el seu propi bucle infinit de control, com *Kivy*, *PyQt4* o *Twisted*, les coses es compliquen. Al compartir estància en el mateix codi més d'una d'aquestes llibreries, cadascuna d'elles vol tenir el control del flux de la execució, és a dir, a portar les rendes dels esdeveniments que hi tenen lloc, que és de fet el que estan acostumats a fer quan estan sols.

Per aconseguir la interacció entre llibreries els propis desenvolupadors faciliten altres llibreries i mòduls complementaris i d'una extensió no massa elevada que permeten unir-les, donant la prioritat dels esdeveniments a qui toqui i pensant com fer-ho tot. Aquestes llibreries s'han d'instal·lar i després cridar correctament.

Les llibreries que s'han hagut d'unir i per tant que s'ha hagut d'esbrinar com fer-ho han estat *Kivy* amb *Twisted* per una banda i *PyQt4* amb *Twisted* per altre.

6.6. Altres llibreries

Hi ha altres llibreries que de caràcter més breu i senzill que s'han utilitzat en el treball. Totes aquestes llibreries ja es troben incorporades a Python de manera estàndard i per tant no es requereix la seva instal·lació. Aquestes llibreries són *copy, math, string, random, ast, functools, os, platform, ctypes, shutil* i *sys.*

7. El joc

El mini joc creat per provar l'arquitectura i les llibreries plantejades té per nom **StarJet** i consisteix en la realització de petites batalles de naus espacials entre els diferents jugadors que comparteixen una partida online. Cada jugador controla una nau per un mateix planeta de petita mida i ha de aconseguir trobar i matar als enemics abans que el matin a ell.

Fig. **7.1.** Imatge promocional de l'aplicació StarJet

Els controls dels que disposen els jugadors són una palanca de comandament (*joystick*) que es pot controlar amb un dit, i un botó de dispar que es pot controlar amb l'altre. Els jugadors han de controlar amb el mateix *joystick* la direcció en la què volen moure's així com la propulsió que volen donar a la nau segons la distància amb què separin la palanca. Els jugadors podran veure la vida o energia que tenen en cada moment i podran veure un mini-mapa que els mostra en tot moment la posició en què es troben en el planeta.

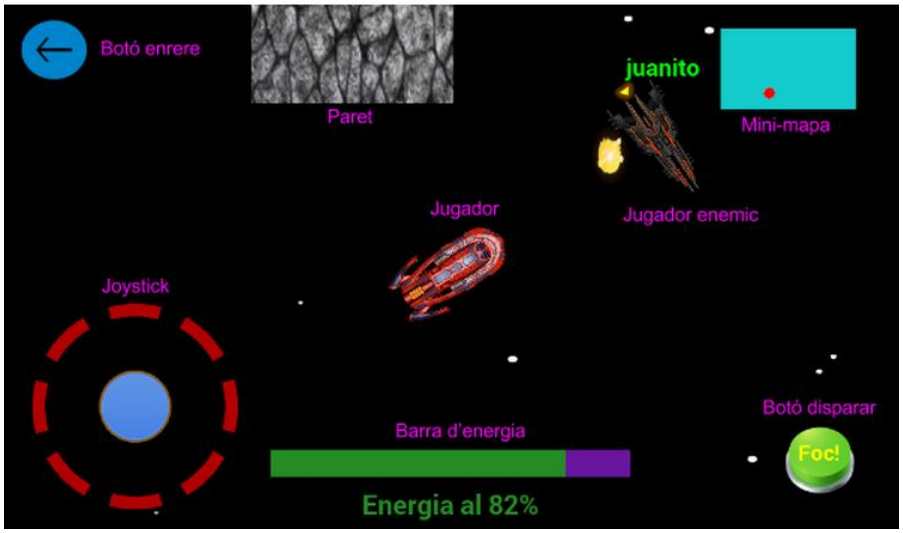

Fig. **7.2.** Escenari d'una partida amb notes dels controls i elements

7.1. Visualització

A continuació es mostra la visualització del joc amb les diferents finestres que conté:

Menú principal

El menú és la finestra principal del joc i és la finestra en la què apareixen els jugadors tot just al obrir l'aplicació, però no podran desenvolupar cap acció fins que tinguin connexió amb el servidor i fins que hagin validat el seu compte. Uns missatges emergents (*popups*) impedeixen que l'usuari accedeixi sense autorització mostrant el missatge corresponent.

Fig. **7.3.** Finestra menú principal de l'aplicació StarJet un cop s'ha iniciat sessió

Si s'ha aconseguit establir una connexió i iniciar sessió, l'usuari podrà accedir a les diferents finestres de l'aplicació mitjançant el botó corresponent. Aquestes finestres són els ajustaments, els crèdits, la pràctica per un sol jugador, i la sala multi-jugador des de Juga. Apart d'això l'usuari pot triar tancar sessió amb el boto Compte i pot triar la nau que pot fer servir per jugar clicant sobre el dibuix corresponent de les 6 naus que hi ha a la finestra. També es podrà sortir de l'aplicació clicant el botó enrere del teclat del mòbil Android.

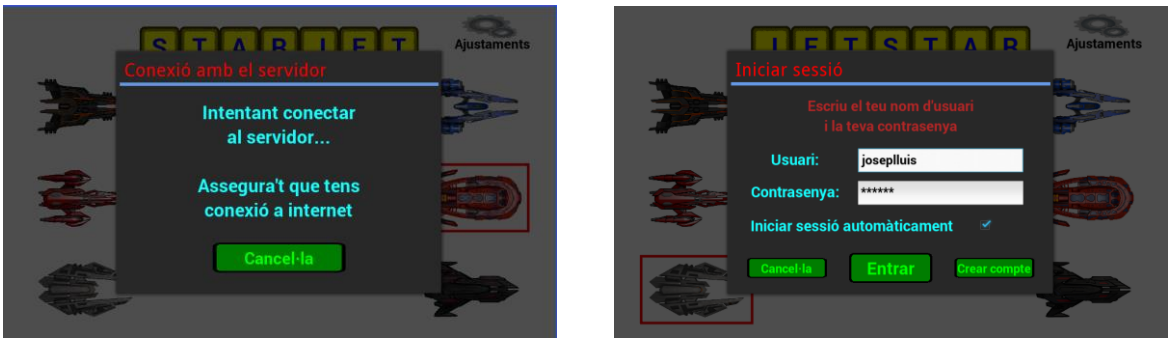

Fig. **7.4.** *Popups* de connexió i d'inici de sessió al menú principal

Sala multi-jugador

Aquesta és la sala que en la què apareixen els usuaris quan volen començar una partida online contra altres jugadors. El que han de fer és senzillament prémer el botó de "Buscar jugadors" i automàticament s'afegiran a la llista d'espera. Aquesta llista es troba al servidor i mirarà quins jugadors emparella amb quins per començar a jugar. Un cop ho hagi fet esborrarà ambdós jugadors de la llista i els permetrà començar la partida. Si un jugador abandona la partida apareixerà en aquesta sala, però no estarà buscant partida ni afegit a la llista d'espera, en canvi, els jugadors amb els què estava jugant, sí que hi apareixeran.

A la figura 7.5 es pot veure la sala multi-jugador del jugador amb nom d'usuari **prova1,** amb 0 victòries, que ha estat emparellat amb el jugador **prova7,** amb una victòria.

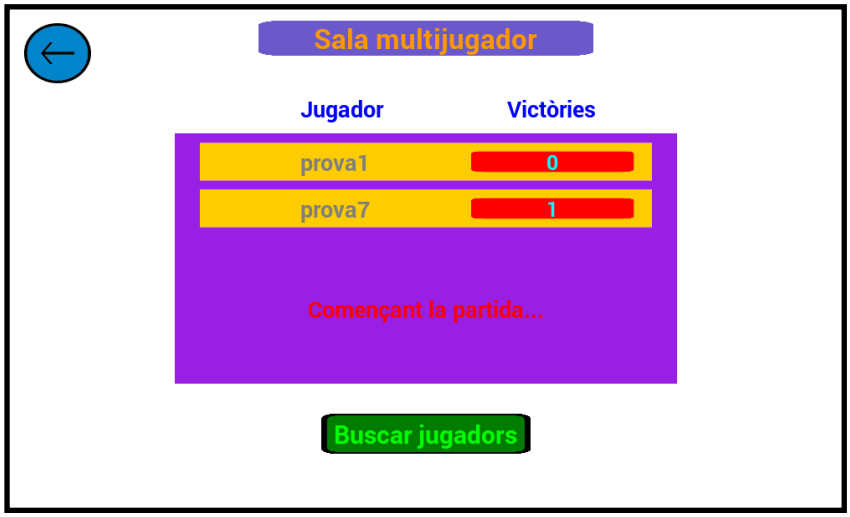

Fig. **7.5.** Finestra sala multi-jugador de l'aplicació StarJet

En aquesta finestra es podria crear un complex algorisme per a la creació de partides que tingués en compte el número de persones que hi ha a la llista d'espera i el número de victòries que tenen. Amb aquest mètode es podria emparellar als jugadors amb altres usuaris que tinguin victòries semblants per fer les partides més justes fent que tothom jugués amb persones amb nivell semblant.

Tot i tenir aquesta idea l'algorisme d'emparellament de jugadors actual no és més que la unió dels 2 primers jugadors que es troben a llista d'espera. Es van emparellant de 2 en 2 sempre i quan hi hagi més d'una persona esperant. Això s'ha fet així perquè de moment només es tracta d'un prototip del joc i pel moment no s'espera que hi juguin gaires persones al mateix temps.

Joc

Aquesta finestra és l'escenari del joc, i hi haurà dos d'elles, una pel mode multi-jugador i una per a la pràctica. Les dos finestres són instàncies de la mateixa classe de Python i tenen algunes petites variacions en alguns dels seus mètodes i atributs com per exemple en l'enviament de dades al servidor, que en el mode de pràctica no s'enviarà res.

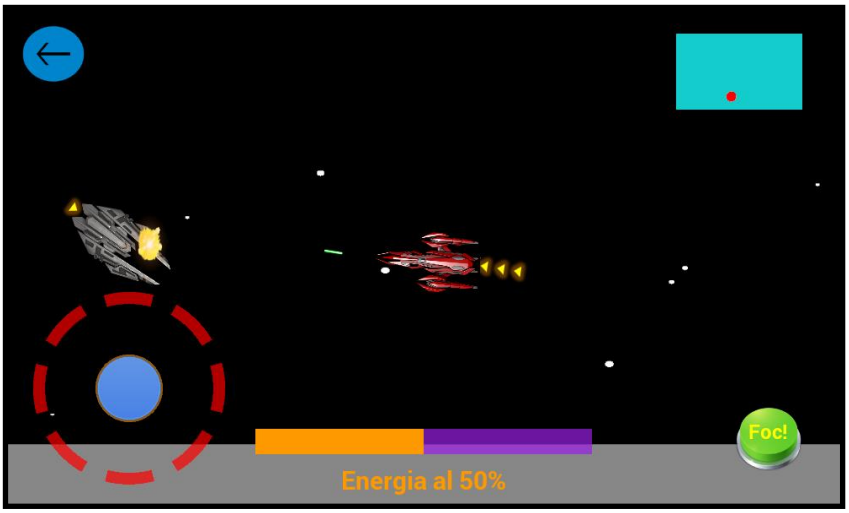

Fig. **7.6.** Finestra joc de l'aplicació StarJet en el mode multi-jugador

En ambdós finestres el jugador apareixerà aleatòriament en un dels quatre cantons del planeta i podrà moure's lliurement. En el cas de la pràctica el jugador estarà sol i només podrà provar controls, però en el cas multi-jugador tots els jugadors que formin part d'una mateixa partida apareixeran en cantons diferents i s'hauran de buscar i eliminar-se. Tot i el que el codi del joc està preparat per acceptar el número de jugadors que es vulgui en aquest prototip només s'ha contemplat la opció de fer partides de 2 jugadors, definint qui guanya i qui perd al acabar aquesta partida. El que detecta el joc és si el propi jugador perd. Si la seva barra d'energia és igual o inferior a 0 haurà perdut i ho comunicarà al servidor, que a la vegada li dirà a l'altre jugador que ha guanyat. Tot just acabar la partida els jugadors ja poden començar-ne una nova ja que apareixeran a la finestra sala multi-jugador.

Com s'ha vist a la finestra sala multi-jugador, els jugadors es qualifiquen amb un número de victòries, com més victòries té un jugador, més ha jugat i teòricament millor jugador és. Al acabar una partida a cada jugador se li suma una unitat a victòries o a derrotes, i la base de dades ho emmagatzema. De la mateixa manera que al acabar una partida s'actualitza el número de partides jugades, en el moment que s'inicia sessió es rep del servidor el número de victòries i derrotes que té l'usuari, i com el servidor té el control no es poden fer trampes.

Dins d'una partida en mode d'un jugador o en mode multi-jugador, els esdeveniments que hi tenen lloc es refresquen 24 cops per segon (24 FPS). Això es dona a terme repetint una sèrie de funcions periòdicament variant certs paràmetres. Aquestes funcions serien les actualitzacions dels paràmetres i elements del joc, com la velocitat, la rotació, la posició, la posició dels focs de la nau, la posició de les bales, la barra d'energia, les col·lisions amb les parets i les col·lisions de les bales amb parets i altres jugadors.

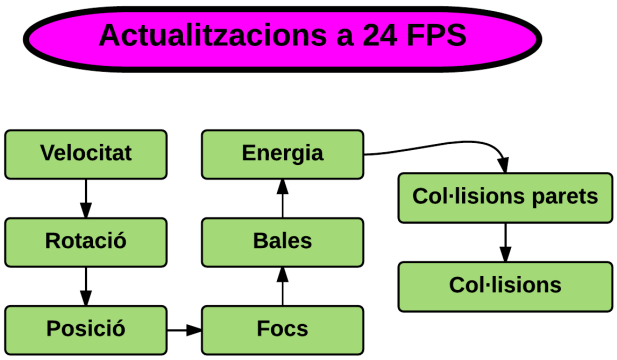

Fig. **7.7.** Esquema de l'actualització de les diferents funcions del joc a 24 FPS

- **Velocitat:** S'actualitza la velocitat vectorial a la que va el jet tenint en compte la rotació a la que s'està anant i comprovant que no es superi la velocitat límit i si el jugador està accelerant o no. També s'actualitza la velocitat tenint en compte la fricció amb el planeta i la gravetat en la direcció **y** i sentit cap a baix.

- **Rotació**: S'actualitza la rotació de la nau. La nau s'ha de dirigir en la mateixa direcció a la què apunta el *joystick* però ho fa de manera progressiva i no instantània. La nau gira automàticament pel camí més curt fins a trobar la mateixa direcció. Això ho fa fent un simple càlcul d'un producte vectorial en 2 dimensions.

- **Posició**: S'actualitza la posició del mapa. Com la posició de la finestra del mòbil no es pot moure (render), el que es fa és moure la imatge que correspon al planeta amb tots els seus elements. Això és molt ineficient però no s'ha trobat cap manera millor de fer-ho al tractar-se d'una aplicació amb un escenari major que la mida del mòbil. L'únic que s'ha de fer és moure el planeta en comptes de la nau invertint el signe de moviment. Això carrega molt les operacions a realitzar i s'intentarà millorar.

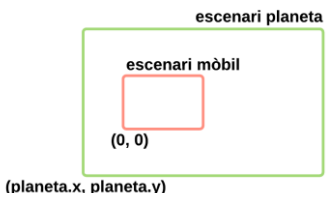

Fig. **7.8.** Estructura de l'escenari del mòbil i l'escenari del planeta

- **Focs**: S'actualitzen la posició dels focs de la nau així com el número de focs que hi ha visible. Les posicions depenen de la rotació de la nau i el número de focs depèn de l'acceleració que s'està aplicant a la nau. Al haver-hi 3 focs es mostraran tots 3 a partir dels 2/3 de la propulsió màxima que es pot aplicar, 2 focs a partir de 1/3 i 1 foc a propulsions menors de 1/3 i que no sigui 0, que en tal cas no es mostrarà cap foc.

- **Bales**: S'actualitzen les posicions de les bales del propi jugador i de les bales dels enemics. Les bales del jugador es creen cada 0.5 segons que es té premut el botó de disparar i apareixen a la posició a la què es troba la nau i surten amb la seva rotació. Les posicions que van tenint les bales un cop s'han creat han de ser absolutes i no relatives, ja que ara la bala forma part del planeta i qualsevol altre jugador l'ha de veure. Això s'aconsegueix definint que les seves posicions de les bales depenguin de la posició a la que es troba el planeta, que depèn de la velocitat de la nau, i d'aquesta manera si es disparés una bala amb la mateixa velocitat que a la que va la nau, ambdós objectes es veurien quiets.

- **Energia**: S'actualitza la barra d'energia de la nau i es va comprovant el seu valor tot canviant de color la barra quan passa de certs punts. Es comprova si l'energia ha arribat a 0 i s'actua en tal cas acabant la partida i mostrant una gran explosió a la posició a la què es troba la nau.

- **Col·lisions parets**: Es tenen en compte en tot moment les possibles col·lisions de la pròpia nau amb les diferents parets del planeta. En cas de col·lisió amb una paret es restarà una mica d'energia a la nau i se li aplicarà un rebot fent una reflexió al seu vector velocitat, amb la normal de la paret. També s'invertirà el sentit del vector velocitat ja que l'algorisme de la reflexió en sí no ho fa.

- **Col·lisions**: Es processen totes les altres col·lisions corresponents a les bales. Es processen les col·lisions de les bales amb les parets i de les bales amb les altres naus enemigues si és que és la finestra multi-jugador. També es processen les bales dels enemics amb la pròpia nau del jugador. En qualsevol d'aquests xocs es realitza el so de cop corresponent i es mostra una explosió de petites dimensions a la posició de l'impacte.

A part de tots aquests elements també es fan altres coses com l'actualització de la posició de la nau en el mini mapa realitzant una simple proporció. També es processa en tot moment l'esdeveniment de clicar amb la pantalla i agafar el *joystick*. Aquest esdeveniment només detecta quan el jugador ha clicat a la posició en què es troba la palanca de comandament i indica que s'ha de començar a girar.

Crear compte

Finestra accessible des de la *popup* d'inici de sessió al menú principal només quan no s'ha validat el compte. Els jugadors poden accedir a aquesta finestra per crear-se el seu compte amb un nom d'usuari i una contrasenya. La contrasenya s'haurà de repetir per assegurar-se que s'escriu correctament.

Fig. **7.9.** Finestra crear compte de l'aplicació StarJet

Aquesta finestra inclou dos passos de validació del compte. El primer pas té lloc a la pròpia aplicació i comprova que el jugador hagi escrit un nom d'usuari i una contrasenya correctes a nivell de caràcters i longitud. Per ser correctes s'han de complir uns certs requisits:

- La longitud del nom d'usuari s'ha de trobar entre 1 i 12 caràcters.
- La longitud de la contrasenya s'ha de trobar entre 4 i 12 caràcters.
- El nom d'usuari i la contrasenya no poden contenir espais en blanc.
- El nom d'usuari i la contrasenya han de contenir només caràcters permesos. Aquests caràcters són només lletres i números i s'han utilitzat els mètodes *letters* i *dígits* del mòdul *string* de Python per fer la comprovació.
- Els camps de contrasenya i repeteix contrasenya han de contenir el mateix text.

El segon pas de validació té lloc al servidor un cop s'ha completat el primer pas. Aquest pas comprova si el nom d'usuari escollit està lliure i no existeix per tant a la base de dades. Si es compleix això es crearà el compte i l'usuari ja podrà iniciar sessió. Una persona es pot crear tants comptes com vulgui sempre i quan estiguin disponibles i li siguin de utilitat.

Ajustaments

Finestra accessible des del menú principal en què l'usuari pot modificar alguns aspectes de configuració del joc com certs ajustaments de música i altres. Aquests paràmetres es guardaran en un fitxer de configuració de text en el propi mòbil i es carregaran cada cop que s'obri l'aplicació de manera que sempre es trobaran tal i com es van guardar.

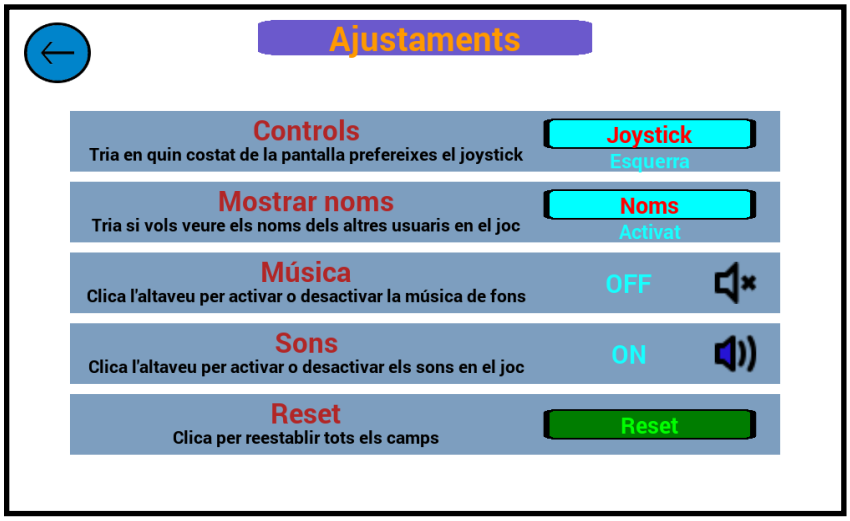

Fig. **7.10.** Finestra ajustaments de l'aplicació StarJet

Les opcions a modificar són:

- **Controls**: Opció que permet triar el costat de la pantalla en què es troba el *joystick* i el botó de dispar intercanviant-los entre ells.
- **Mostrar noms**: Opció que permet triar si es vol o no veure els noms dels jugadors sobre les naus mentre es realitzen partides multi-jugador.
- **Música**: Opció que permet activar o desactivar la música de fons de l'aplicació.
- **Sons**: Opció que permet activar o desactivar els sons que té l'aplicació mentre es juga.
- **Reset**: Opció que restableix tots els anterior camps al seu valor inicial.

Crèdits

Finestra accessible des del menú principal que permet veure informació del creador de l'aplicació així com dels agraïments.

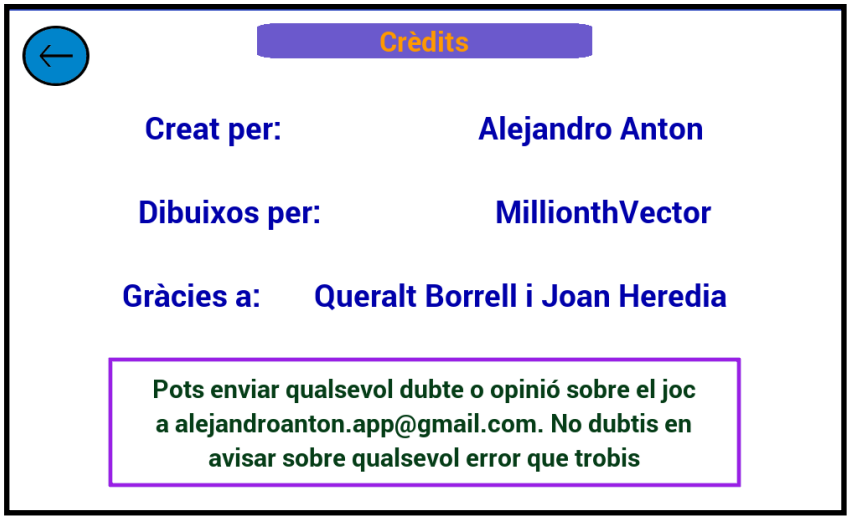

Fig. **7.11.** Finestra crèdits de l'aplicació StarJet

La finestra també inclou un petit requadre amb text cap al públic informant de l'email del desenvolupador per enviar qualsevol dubte o opinió que tinguin sobre el joc així com avisar sobre qualsevol error que puguin trobar.

7.2. Funcionament general

El funcionament de l'aplicació consisteix en el moviment de l'usuari entre les diferents finestres del joc que s'han vist a l'apartat anterior. Un cop iniciada l'aplicació el joc intenta connectar amb el servidor. Si ho aconsegueix demanarà autentificació a l'usuari, que haurà de validar el seu compte si ja en disposa d'un o registrar-se primer si no en té. Si no s'aconsegueix la connexió ja sigui perquè l'usuari no disposa d'internet o ja sigui perquè el servidor està apagat, l'usuari no podrà jugar. Tot i que hi ha una part del joc on els jugadors poden practicar ells sols sense necessitat de connexió web, s'ha cregut convenient no permetre l'entrada a menys que el servidor en tingui constància, ja que el joc en sí consisteix en ser multi-jugador. Si en qualsevol moment en què l'usuari té l'aplicació oberta es perd la connexió, aquesta cancel·larà qualsevol cosa que s'estigui fent i tornarà automàticament al menú principal per intentar tornar a connectar-se.

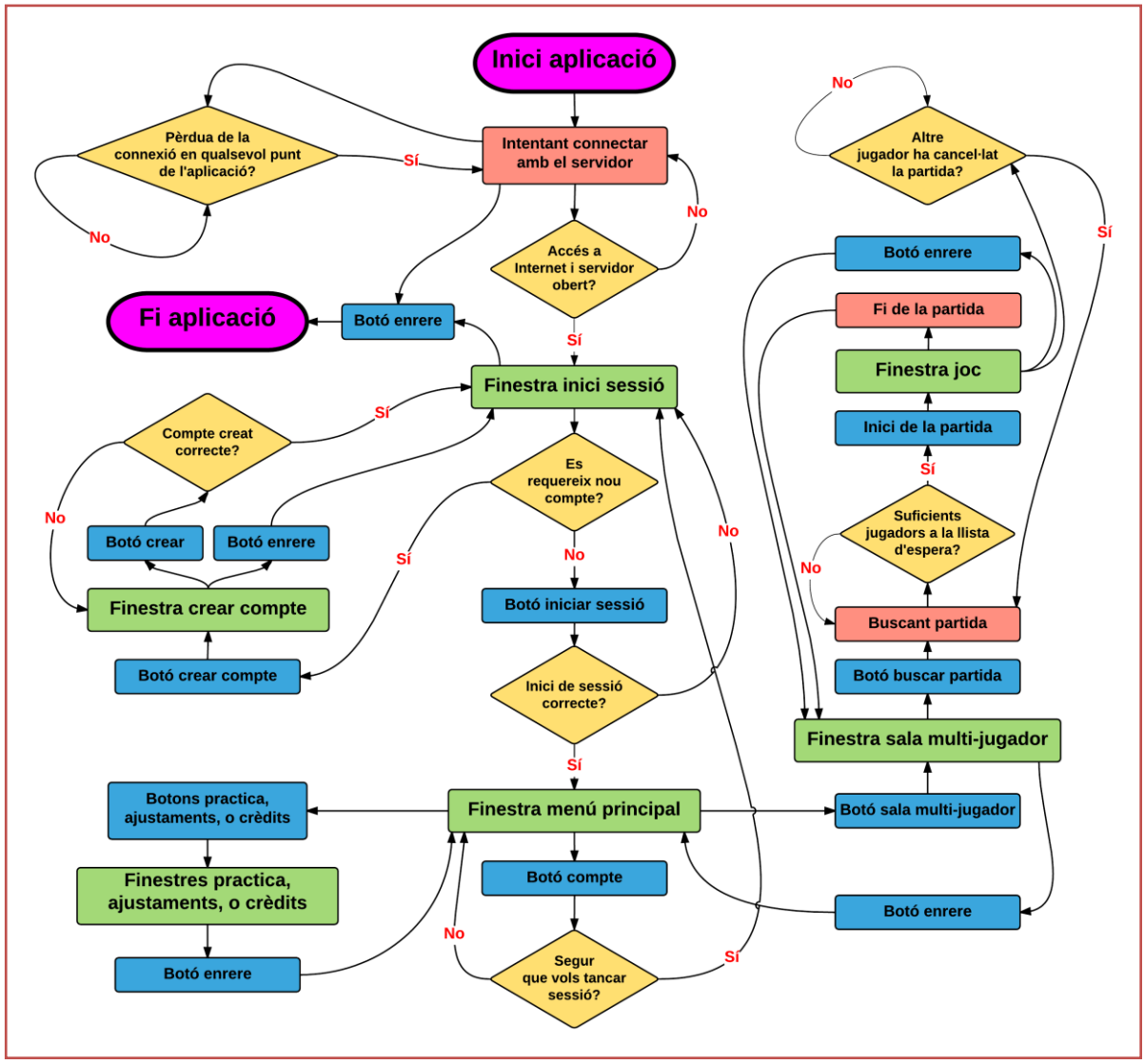

Fig. **7.12.** Diagrama de l'estructura del funcionament de l'aplicació StarJet

7.3. Funcionament multi-jugador

Hi ha diferents tipus d'accions que el joc comunica amb el servidor, com l'addició o sostracció del jugador de la llista d'espera, el moment d'inici de sessió, el moment de la creació d'un nou compte, etc... Aquestes accions es porten a terme enviant una sola dada de text al servidor indicant què s'ha de fer i de quina manera, però en un joc multi-jugador en temps real per actualitzar una partida l'enviament i rebuda de dades s'ha de fer de manera constant entre els diferents jugadors molts cops per segon.

Tots els jocs multi-jugadors es basen en el fet de compartir elements d'un mateix escenari, com la posició en què es troben, la vida que tenen, si corren o caminen, etc... Els diferents jugadors s'envien entre ells les dades que necessiten per actualitzar l'estat de tots els altres jugadors, amb ajuda o no d'un servidor intermediari. La clau d'aquest mètode es troba en el punt d'actualitzar i compartir les mínimes coses possibles amb els altres jugadors.

Per dur a terme una partida online amb altres jugadors en aquesta aplicació es requereix conèixer l'estat que tenen aquests jugadors en cada instant de temps de manera que s'envien constantment dades en format de text al servidor durant la partida i ho fan 10 cops per segon (és a dir a 10 FPS que és cada 100 mili-segons). Aquestes dades que s'envien són rebudes amb un petit retard per l'altre jugador que realitzarà la mateixa tasca d'enviament de dades. A la figura 7.13 es pot veure el procés d'enviament i rebuda de dades que té lloc entre els jugadors cada 100 ms.

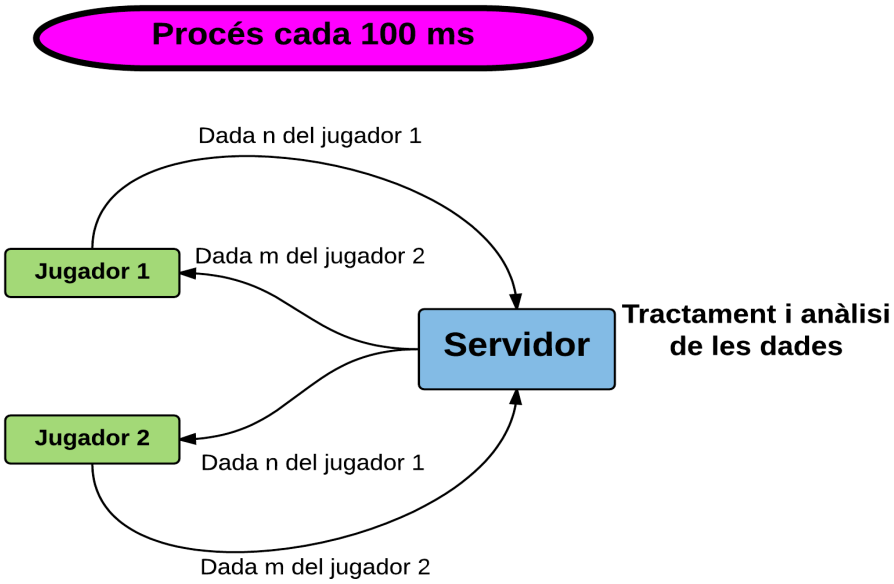

Fig. **7.13.** Esquema del procés d'enviament i rebuda de dades cada 100 ms

Tota dada que s'envia o es rep té una estructura semblant, i és que no són més que strings de Python que contenen la informació a transmetre. Aquesta informació que s'ha de transmetre constantment conté la posició relativa de la nau, la seva rotació, l'energia que té i el número de focs del propulsor que està utilitzant. De la mateixa manera que calcula aquests paràmetres i els envia també està pendent de rebre'ls i d'actualitzar la nau enemiga corresponentment. També s'envia sempre el nom del tipus de dada que es tracta.

Hi ha altres dades que s'envien i es reben durant una partida però que es fan de manera asíncrona com un inici de sessió, i són la cancel·lació de la partida si un jugador abandona i el dispar d'una bala. Si es dona una d'aquestes situacions el que es fa és enviar al servidor una dada indicant què ha passat, en el cas d'abandonar només es comunicarà quin jugador ha abandonat i en cas de la creació d'una bala es comunicarà en quina posició i amb quina rotació s'ha creat aquesta per reproduir-la al dispositiu de l'altre jugador.

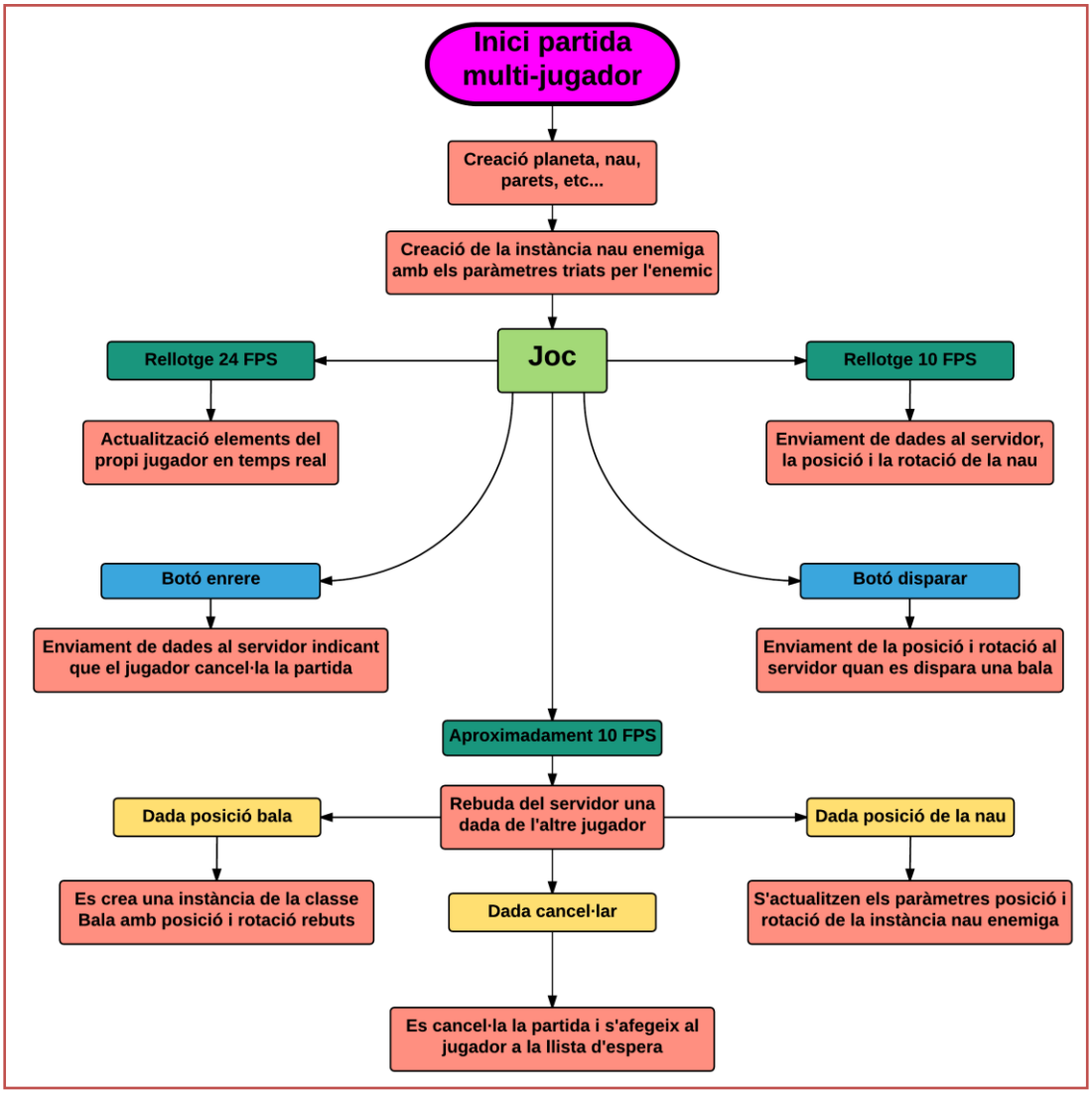

Fig. **7.14.** Estructura dels esdeveniments de comunicació en una partida multi-jugador

Com s'ha esmentat anteriorment és important enviar el mínim d'informació possible. Com s'ha vist el que s'envien són només paràmetres que s'actualitzaran a l'altre dispositiu, si s'ha de crear una classe per representar un jugador, el que es fa en una partida amb altres jugadors es crear una d'aquestes classes per a cada un, però no s'envien entre ells una classe sencera. L'únic que s'envien són els paràmetres de la classe per actualitzar-lo a la classe del mateix jugador de l'altre dispositiu.

Tots els dispositius mòbils tenen mides diferents, i això comporta que tots els elements de l'aplicació s'han hagut de definir amb proporcions relatives i no amb mides absolutes, ja que s'ha de veure igual en tots els dispositius independentment dels píxels que tinguin. Tots els elements del joc estan definits amb una proporció respecte la mida de la pantalla, garantint així que tothom veurà les coses igual.

Durant una partida de StarJet un dels paràmetres que s'envia és la posició, i aquesta posició també ha de ser també relativa i no absoluta, ja que no es coneix quina mida tindrà la pantalla de l'altre jugador. El que es fa durant la partida és enviar la posició relativa dividint l'absoluta entre la mida de la pantalla. Al rebre aquesta proporció a l'altre dispositiu el que farà es fer-la absoluta multiplicant-la per la mida de la seva pantalla. D'aquesta manera les posicions d'ambdós jugadors s'actualitzaran d'una manera coordinada.

7.4. Estructura de fitxers

En aquest apartat es mostren els diferents fitxers de Python en què s'ha estructurat la feina feta i s'expliquen breument. L'estructuració dels mòduls "py." del joc és la següent:

- **main.py**: Mòdul principal que s'encarrega d'engegar l'aplicació i en ell es troba l'estructura que l'organitza. En aquest mòdul es troba la classe principal *App* que és la que s'instanciarà i es cridarà per iniciar l'aplicació. Aquesta classe s'encarrega de fer diferents tasques com la inicialització de la finestra principal, la càrrega del fitxer de configuració, la connexió amb el servidor i l'administració de les connexions i dades rebudes. En aquest mòdul es troba per tant la classe *GameLayout* que és la finestra principal de l'aplicació. Aquesta finestra serà l'administradora de totes les diferents finestres del joc i s'encarregarà de posar la que toqui en el moment que toqui. També és la classe que s'encarrega d'administrar qüestions com els sons i la música així com d'altres coses com la configuració del teclat d'Android. Per últim en aquest mòdul es troben les classes *Connexio* i *Factoria* que són les classes del mòdul *Twisted* que permeten treballar amb les tasques de la comunicació web.

- **widgets.py**: Mòdul de llarga extensió que conté cadascuna de les classes que corresponen a les finestres que té l'aplicació (heretades de la classe *Widget*). Cada finestra del joc té la seva classe en aquest mòdul i cada classe té tots els elements i funcions que la finestra necessita. Aquestes funcions van des de la inicialització de tots els elements de la interfície fins a les accions que s'han de fer al clicar cada botó. Hi ha funcions característiques de cada finestra com les actualitzacions periòdiques d'elements de la classe de la finestra Joc. Totes aquestes classes són importades i inicialitzades per la classe *GameLayout* del mòdul *main.py*.

- **elements.py**: Mòdul que conté un seguit de classes d'extensió mitja i curta que corresponen a diferents elements que s'afegeixen a certes finestres, especialment al joc. Aquests elements serien les naus, els planetes, o les bales entre altres i tots ells són cridats per la classe finestra que les requereixi del mòdul *widgets.py*.

- **funcions.py**: Mòdul que conté tot un seguit de funcions que es s'importen a cadascun dels altres mòduls que les requereixen. Aquestes funcions són càlculs vectorials, distàncies, tractament de vectors, obtenció de colors, etc...

- **variables.py**: Mòdul que conté la classe *Variables* que conté totes les variables de caràcter general que es fan servir tant a l'aplicació principal com a cadascuna de les classes i funcions. Aquesta classe s'instanciarà un cop a la classe *App* del mòdul *main.py* i es passarà com a paràmetre als altres mòduls que la necessitin.

A la figura 7.15 es pot veure un diagrama de l'estructura dels diferents mòduls que s'han mencionat amb els elements que contenen. Les fletxes que surten d'un element van a parar a tot aquell objecte que aquest element crida en ell mateix. L'execució del programa comença a la classe *App* del mòdul *main.py*.

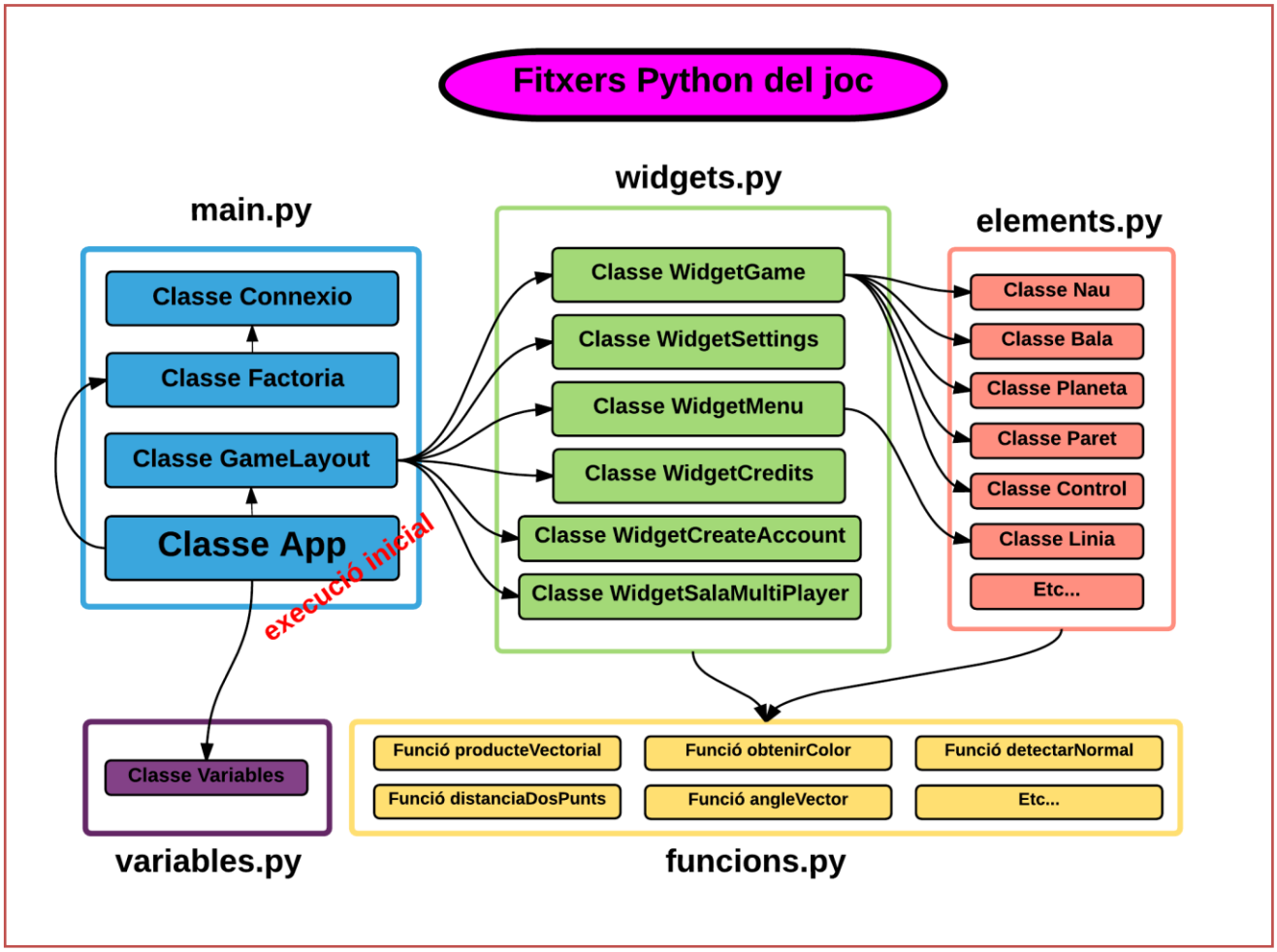

Fig. **7.15.** Estructura dels fitxers del joc amb els seus elements i fletxes.

7.5. Problemes del prototip

Hi ha certes qüestions del prototip amb les que s'ha estat treballant durament i amb les que s'han trobat grans problemes que encara ara es tenen, i que s'intentaran arreglar i millorar en un futur.

El primer problema que presenta aquest prototip de joc és el fet que depenen del dispositiu en què s'estigui executant el joc, els rellotges que actualitzen la informació del joc per unitat de temps varien. Això significa que com menys potent és un dispositiu més li costa mantenir el ritme establert pel rellotge d'actualització de paràmetres. S'ha pogut comprovar que amb la tableta Samung, que té més de 4 anys, el rellotge baixa dels 24 FPS depenen de si es fan moltes accions a la vegada (12 FPS a vegades), amb el mòbil s'acostuma a mantenir als 23, i a l'ordinador sempre es troba a 24. Aquest fet no seria del tot important si es tractes d'un joc d'un jugador, però al tractar-se d'un joc multi-jugador és vital que tothom actualitzi els paràmetres de la mateixa manera. Aquest fet s'ha intentat corregir d'una manera poc professional intentant augmentar els FPS amb una relació entre els FPS actuals i els FPS que hauria de tenir (24). El que s'ha d'intentar fer en un futur és aconseguir un rellotge síncron universal, fent que alguns paràmetres com la posició i la velocitat no variïn amb els FPS si no per unitat de temps transcorreguda. El fet que a alguns dispositius els hi costi aguantar el ritme del rellotge pot ser degut a l'excés de càlculs a realitzar. Això ja es va pensar i no només es va baixar el nombre de FPS al que s'executen les operacions si no que es va simplificar dràsticament les operacions a fer.

Un altre problema que presenta el joc és que moltes aplicacions per mòbil, i sobretot en 2D i sobretot les de *Kivy*, no estran preparades per treballar amb escenaris de mides majors que la mida de la pantalla del mòbil. El que es vol dir amb això és que si es tractés d'un joc 3D el que s'acostuma a fer és dissenyar un escenari i moure la càmera a allà on es vulgui situar la vista i els elements del joc. El problema de *Kivy* és que no permet moure la posició del *render*, és a dir, la posició del que veu la pantalla. El que s'ha hagut de fer en comptes de moure la càmera és moure l'escenari sencer, i això apart de ser una mica una bestiesa afavoreix al problema anterior de què s'executen molts càlculs per unitat de temps. S'han de moure constantment una gran quantitat d'objectes que el jugador no veu i que no faria falta que s'estiguessin processant en tot moment. Per tant s'hauria d'aconseguir trobar una solució per aquest problema o aconseguir que només es processi la part de l'escenari que toca, que amb *Kivy* no és fàcil.

8. El servidor

El servidor és l'aplicació que s'encarrega d'administrar totes les connexions i partides de les persones que volen accedir al joc. Ha estat creat utilitzant *Twisted* i *PyQt4* i ha d'estar encès i funcionant en tot moment a l'ordinador que actua com a servidor. El servidor com a tal no requereix la interfície gràfica per funcionar ja que el codi en sí no ho necessita, tot i això, molts cops és millor disposar d'interfície d'usuari per a manipular i veure l'estat de l'aplicació mòbil general d'una manera més fàcil i visual.

8.1. Visualització

El servidor només té una finestra a la interfície gràfica que es manté durant tota l'execució.

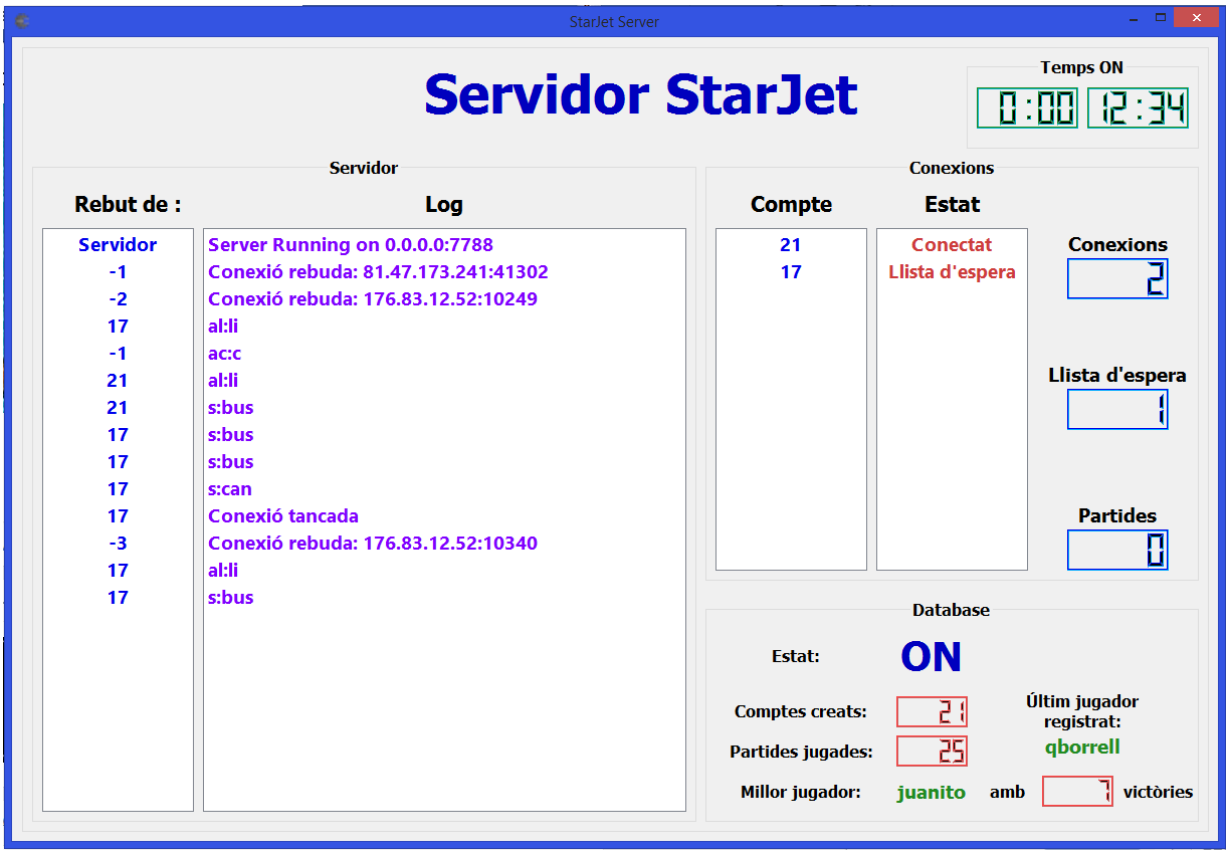

Fig. **8.1.** Visualització genèrica de la interfície d'usuari del servidor StarJet

En aquesta finestra principal del servidor s'actualitzen en temps real diferents paràmetres i elements del joc i de la base de dades, que s'explicaran a continuació.

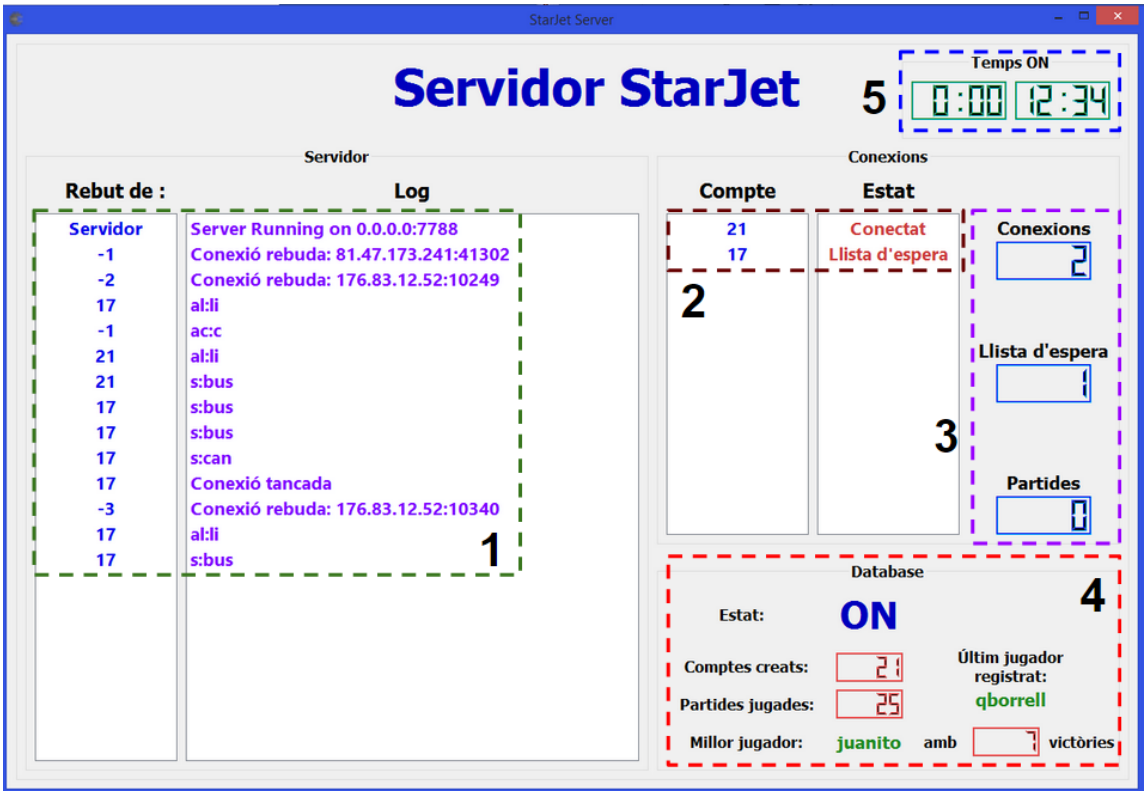

Fig. **8.2.** Visualització de la interfície d'usuari del servidor StarJet amb els seus elements

- **Punt 1**: En aquest punt es poden trobar dues llistes que corresponen a l'enumeració dels diferents esdeveniments que van tenint lloc en el servidor provinents de tots els dispositius que es connecten i del propi servidor. Aquests esdeveniments simbolitzen tots els moments en què un jugador ha de comunicar-se amb el servidor per accedir o guardar una dada i corresponen a moments com l'inici i tancament de connexions així com l'accés a la base de dades i l'inici de sessió. A la llista de la dreta es veu el text de l'esdeveniment que ha tingut lloc i a la llista de l'esquerra el propietari del missatge, sigui el servidor o un jugador que ha accedit, que se'l reconeix amb el seu número de compte i no amb el nom d'usuari. Si apareixen números negatius significa que l'usuari ha obert l'aplicació i disposa de connexió però encara no està assignat a cap compte perquè de moment no ha iniciat sessió.

- Punt 2: Les dues llistes del punt 2 corresponen a tots els jugadors que hi ha actualment connectats al servidor. A la llista de l'esquerra indica quin número de compte tenen els usuaris connectats i a la dreta l'estat amb què es troben. Cada usuari que hagi iniciat sessió pot tenir tres estats diferents, connectat, llista d'espera, o en partida. Cada estat significa que l'usuari es troba fent el què l'estat indica.

- Punt 3: Aquests 3 *displays* de nombres de 7 segments només comptabilitzen el nombre d'usuaris que es troben connectats amb el servidor i quants d'ells es troben en llista d'espera i quants d'ells es troben en partida. Tots tres estats mencionats anteriorment signifiquen que el jugador està connectat, ja que si no apareixeria al servidor, però l'estat connectat vol dir que no es troba en cap dels dos altres i que té l'aplicació oberta i no està jugant ni esperant per jugar.

- Punt 4: Aquest punt correspon a la part de la base de dades. En aquest requadre es pot trobar informació de la connexió del servidor de Python amb el servidor MySQL i la base de dades del joc. Aquesta informació indica quants comptes hi ha creats i quantes partides s'han fet en total per tots els jugadors. També es pot veure quin és l'últim usuari que s'ha registrat i el millor jugador amb el seu número de victòries.

- Punt 5: El punt 5 només indica mitjançant 2 *displays* de nombres 7 segments el temps que el servidor porta obert.

account	username	password	wins	loses
1	ac4	hola1	0	0
2	oleole	hola2	1	1
3	yiha	caci	0	0
4	bona	ca22	2	3
5	ale1	ca22	1	2
6	oleeeeeeeeee	cccc	0	0
7	ieee	cccc	3	1
8	aaaaaaaaaaa	vvvvvvvvvv	0	0
9	yiha2	sdsd	0	0
10	ca44	ca44	0	0
11	koke	neme	0	0
12	juanito	ole1	7	1
13	prova1	prova1	2	5

Fig. **8.3.** Primers 13 comptes de prova de la taula d'usuaris de la base de dades starjet

8.2. Funcionament general

L'execució de l'aplicació servidor comença inicialitzant tots els elements de la interfície gràfica així com inicialitzant també la connexió amb la base de dades MySQL. Tot seguit, s'obre el que realment és el servidor, és a dir, l'aplicació comença a escoltar per a l'arribada de connexions. La seva feina principal és afegir al programa tota connexió que li arribi i tractar totes les dades que li arribin de totes les connexions que tingui guardades.

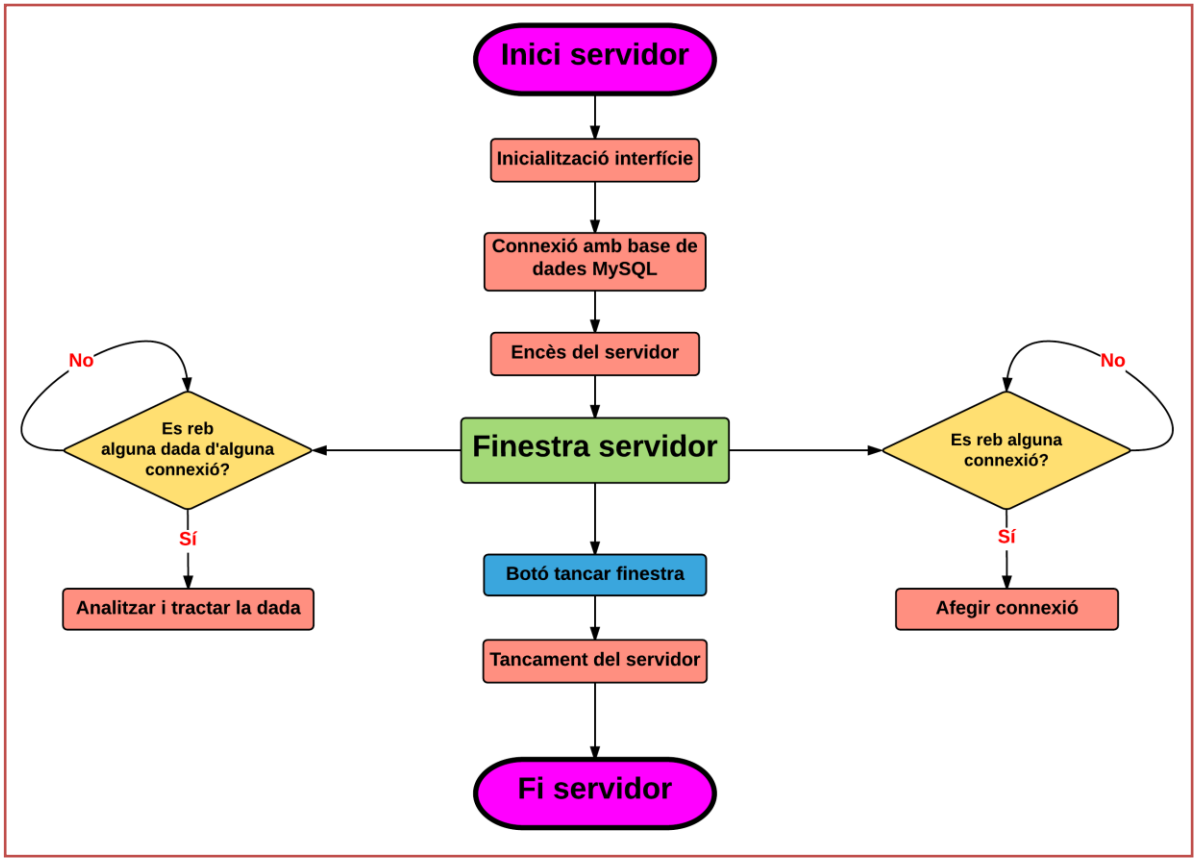

Fig. **8.4.**. Diagrama de l'estructura del funcionament del servidor StarJet

Tot just es rep una connexió s'associa al usuari d'aquesta connexió un número d'identificació per saber amb qui s'està comunicant. Aquest número d'identificació és negatiu i s'actualitzarà amb el número de compte un cop l'usuari hagi iniciat sessió i és negatiu per no confondre'l amb el número de compte. Una connexió s'eliminarà si el jugador surt de l'aplicació ja que saltarà la eventualitat de pèrdua de connexió del servidor, i s'eliminaran automàticament totes les connexions si el servidor es tanca.

8.3. Funcionament multi-jugador

Totes les dades que es reben es tracten i a continuació es torna a enviar un missatge amb la resposta corresponent si és necessari, ja que algunes dades no requereixen respondre al mòbil. Hi ha dades que no requereixen paràmetres i que només avisen del que ha passat, però d'altres contenen paràmetres que es transmeten entre els jugadors com la posició i la rotació de les naus en una partida online. Totes les dades no són més que strings de Python que es creen per enviar-les i que es descodifiquen i es tracten quan es reben.

A la figura 8.5 es pot veure l'estructura que tenen totes les dades que s'envien amb dos exemples d'elles, una amb paràmetres i una sense. El primer exemple té de tipus 1 "s" que indica sala multi-jugador, i de tipus 2 "can", que significa cancel·lar. Aquesta dada s'envia des del mòbil quan un jugador ha abandonat la sala d'espera i no requereix respondre al usuari. El segon exemple té de tipus 1 "g" que significa "game", i de tipus 2 "pos", que significa posició. Aquesta dada és exactament del format de les que s'envien entre els jugadors d'una partida cada 100 mili-segons. Com s'ha explicat mentre es descrivia el joc, els paràmetres que s'envien en cada dada apart dels tipus són la posició relativa de la nau (0.48, 0.23), la rotació de la nau (124.3º), l'energia del jet (78), i el número de focs que està fent servir (2). Aquestes dades les utilitzarà l'altre jugador per actualitzar l'estat del seu joc.

```
dada = "tipus1:tipus2:param1:param2:param3:paramN"

dadaExemple1 = "s:can"
dadaExemple2 = "g:pos:0.48:0.23:124.3:78:2"
```

Fig. **8.5.** Estructura i exemples de les dades que s'envien servidor i mòbil

Hi ha diferents cops en què el servidor i el mòbil es comuniquen i tots ells es poden veure a la taula de la següent figura 8.6. En aquesta taula es poden veure les diferents dades que s'envien amb una petita descripció, els seus tipus, els seus paràmetres, i si el servidor ha de respondre o no a la dada que li arriba. A continuació s'explicaran breument:

Dada	Tipus1	Tipus2	param1	param2	param3	param4	param5	Resposta?
Actualització paràmetres partida online	g	pos	posició x	posició y	rotació	energia	num focs	sí
Creació bala	g	bal	posició x	posició y	rotació			sí
Cancel·lació d'una partida	g	eli						sí
Fi d'una partida	g	fi						sí
Buscar partida a la sala multi-jugador	s	bus	dibuix nau					sí
Cancel·lar buscar partida	s	can						no
Crear compte	ac	c	usuari	contrasenya				sí
Iniciar sessió	al	li	usuari	contrasenya				sí
Tancar sessió	al	lo						no

Fig. **8.6.** Taula que mostra les diferents dades que s'envien amb els seus tipus i paràmetres

- Actualització paràmetres partida online: Dada que s'envien els jugadors 10 cops per segon duran les partides multi-jugador. Ja s'ha mencionat anteriorment els paràmetres que s'envien. Aquestes dades es reenvien als altres jugadors que comparteixen partida.

- Creació bala: Dada que s'envia quan un jugador prem el botó de disparar que serveix per indicar als altres jugadors on i amb quina rotació s'ha de crear una instància d'una bala. Aquesta dada es reenvia als altres jugadors que comparteixen partida.

- Cancel·lació d'una partida: Dada que s'envia quan un jugador abandona una partida amb un altre jugador. Aquesta dada es reenvia als altres jugadors per indicar-los que la partida ha estat cancel·lada i tornen a ser afegits a la sala d'espera.

- Fi d'una partida: Dada que s'envia quan algun jugador ha perdut. Aquesta dada servirà per incrementar en una unitat el número de derrotes del jugador i el número de victòries de l'altre jugador. Aquesta dada es reenvia a l'altre jugador per indicar-li que ha guanyat.

- Buscar partida a la sala multi-jugador: Dada que s'envia quan un jugador prem el botó "Buscar jugadors" de la sala multi-jugador. Aquesta dada afegirà al jugador a la llista d'espera i només s'enviarà una resposta a tots els jugadors si hi ha suficient gent per començar una partida. Aquesta dada conté com a paràmetre el número del dibuix de la nau que el jugador ha triat, d'aquesta manera quan s'iniciï partida es comunicarà als altres jugadors quin és l'aspecte que l'altre jugador ha escollit per actualitzar la classe nau enemiga.

- Cancel·lar buscar partida: S'envia quan un jugador prem el botó enrere des de la sala multi-jugador i no requereix reenviar res perquè no estava assignat a cap partida, senzillament s'elimina al jugador de la sala d'espera que controla el servidor.

- Crear compte: Dada que s'envia quan un jugador vol crear-se el compte i ja ha complert amb el primer pas de validació intern a l'aplicació. Aquesta dada conté el nom d'usuari i la contrasenya i comprovarà que el nom d'usuari no existeixi. Reenviarà com a resposta al propi jugador si el seu compte ha estat validat o no.

- Iniciar sessió: Dada que s'envia quan un jugador inicia sessió i conté el nom d'usuari i la contrasenya. El que farà el servidor es comprovar si la contrasenya emmagatzemada a la base de dades amb el nom d'usuari enviat és la mateixa que la contrasenya que el jugador ha enviat. Reenviarà al propi jugador sí l'inici de sessió ha estat validat o no, i si ho ha estat el permetrà iniciar sessió i començar a jugar.

-Tancar sessió: Dada que s'envia si un jugador tanca sessió. Aquesta dada només servirà per informar al servidor que un usuari segueix connectat però no té cap compte assignat.

8.4. Estructura de fitxers

La estructuració resultant dels mòduls "py."del servidor és la següent:

- **main.py**: Mòdul principal que s'encarrega d'engegar l'aplicació i en ell està tot el cos del servidor amb totes les funcions que fan possible la comunicació i administració de comandes. En ell es troba la classe *Server* que s'instanciarà per iniciar l'execució del programa i la classe *Connexio* corresponent a les connexions per *Twisted.* També es pot trobar la classe *Game* que no es més que un objecte de Python que simbolitza cada partida que es desenvolupa entre diferents jugadors. La classe *Server* és la classe principal que conté totes les funcions del tractament de dades.

- **sqlmanager.py**: Mòdul que inclou la classe *SQLManager* amb totes les funcions necessàries per accedir a la base de dades MySQL. Compta amb funcions que li permeten connectar-se amb la base i accedir o guardar dades. Aquest fitxer es crida des del mòdul *main.py* per obtenir una única instància de la classe esmentada.

- **mainwindow.py**: Mòdul que inclou les funcions d'anàlisi d'interfície gràfica. Conté la classe *MainWindow* que hereta de la classe *QMainWindow* del mòdul *PyQt4.QtGui.* Aquesta classe s'encarrega de les qüestions d'aspecte i modificació d'elements de la interfície. Aquest mòdul es crida des de *main.py* per obtenir aquesta classe.

- **variables.py**: Mòdul breu que conté una única classe *Variables* amb certes variables com a atributs importants que seran cridats des del mòdul *main.py* per crear una única instància que serà passada als altres mòduls que la requereixin.

- **widget_principal.py**: Mòdul creat a partir de Qt Designer que conté els elements propis que es veuen a la interfície gràfica. Aquest mòdul és cridat per *mainwindow.py* per obtenir d'ell la classe que representa els elements de la finestra.

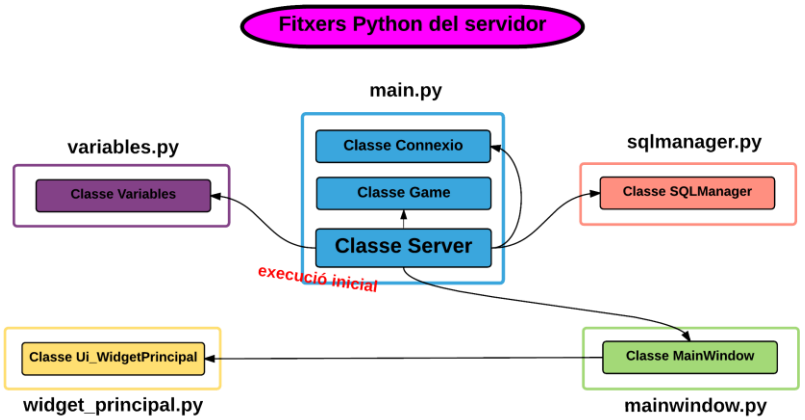

Fig. **8.7.** Estructura dels fitxers del servidor amb els seus elements i fletxes

8.5. Configuració del domini i del port

Per poder desenvolupar una aplicació que funcioni com a servidor és necessari configurar breument la xarxa de què es disposa i tenir clar on s'està executant aquesta aplicació.

L'aplicació servidor s'executa a nivell local a l'ordinador de casa amb un *router* estàndard i s'ha de tenir en compte que cada xarxa té una IP diferent, que pot ser estàtica o dinàmica, i en aquest cas, dinàmica. En un cert dia la IP dinàmica de la xarxa de casa tenia la IP 84.81.102.32, i el programa NoIP DUC s'encarrega d'emmascarar qualsevol valor que tingui la aquesta IP amb el domini *alejandroserver.sytes.net*.

Amb aquest pas s'ha definit el domini al què el servidor funciona i per tant el domini al què els clients s'han de connectar, però a cada xarxa hi tenen accés diferents dispositius com diferents mòbils, tabletes, ordinadors, etc...

El que fa falta especificar a continuació és a quin dispositiu dels què hi ha connectats a la xarxa han d'accedir les connexions per arribar al servidor establert a l'ordinador i això s'indica amb el port, que s'ha d'obrir i configurar. Cada dispositiu connectat a la mateixa xarxa té un altre tipus d'IP que diferencia cadascun d'aquests dispositius dins la mateixa xarxa amb la IP esmentada anteriorment. El que s'ha de fer és accedir a la configuració del *router* i configurar el port al què es vol treballar de manera que estigui lligat amb una d'aquestes IP's, i per tant lligat amb un dispositiu.

A la figura 8.8 es pot veure un esquema de la situació de les IP's en un cert dia, ja que van canviant. Si el servidor es troba a l'ordinador 2 i en aquell dia la IP que tenia 'ordinador era 192.168.1.34 el que s'ha de fer es lligar el port escollit 7788 amb aquesta IP. D'aquesta manera ja es troben el domini i port configurats i l'accés ja està definit i completat.

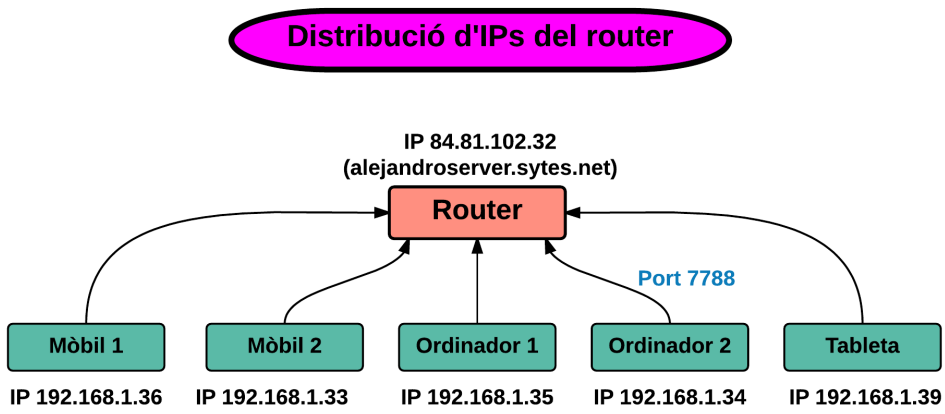

Fig. **8.8.** Esquema orientatiu de la situació de les IP's del router de casa en un cert dia.

9. De Python al mòbil

Haver construït una aplicació amb *Kivy* i visualitzar-la a l'emulador de l'ordinador és molt pràctic per fer proves però el que realment interessa és disposar d'aquesta aplicació al mòbil i penjar-la a la web de manera que la gent pugui accedir a ella i descarregar-se-la.

Per a fer proves al mòbil es pot utilitzar l'aplicació Kivy Launcher però amb això només s'aconsegueix veure com queda l'aplicació gràcies a aquest programa. El que es vol aconseguir és crear una aplicació mòbil independent amb el nom i icona que es desitgi.

Amb tot el material programat en Python s'ha d'escollir la plataforma a la què es vol tenir l'aplicació. *Kivy* ofereix avui en dia la possibilitat de disposar de l'aplicació a Android i a IOS. Tot i que ofereix ambdós possibilitats, Android és una opció molt més senzilla i viable ja que la documentació i eines que existeixen ho fan així. La documentació de conversió a IOS està encara en fase de desenvolupament i el procés és massa llarg i complex, a Android en canvi, tot i que el procés podria ser més breu, és molt més accessible.

Un cop es tenen els fitxers de text ".py" i totes les subcarpetes amb el material necessari per al joc com imatges o sons, el que s'ha de fer es convertir-ho tot a alguna cosa que entengui Android. Totes les aplicacions d'Android funcionen amb uns arxius anomenats APK, que significa que un cop s'aconsegueixi crear aquest tipus d'arxiu ja es pot instal·lar en un dispositiu i funcionar. El que s'explicarà aquí és el mètode de conversió amb la creació de l'APK, i tot seguit, es mostrarà com s'ha penjat l'aplicació al Google Play.

9.1. Creació de l'APK

Per crear l'APK a partir dels fitxers de Python s'ha fet servir un disc virtual d'Ubuntu anomenat *Python for Android* que es pot trobar al web *Kivy* i que inclou unes llibreries mig desenvolupades en Python que poden realitzar aquesta conversió. Es un mètode una mica complicat i embolicat però es pot aconseguir dur a terme. S'ha d'accedir al disc virtual obrint el programa Oracle VM VirtualBox que permet emular el sistema operatiu com una finestra més a l'ordinador, i s'ha de configurar breument. Un cop dins el disc s'han d'executar un seguit de comandes a la terminal per fer i configurar la conversió. Aquestes comandes crearan el fitxer i li afegiran una signatura virtual per identificar al creador. Hi ha dos opcions de fer la conversió, totes elles amb el mateix destí però amb modificacions en el procés, i són el mètode directe per terminal o el mètode mitjançant la llibreria Buildozer. A l'annex s'explica amb detall el procés de conversió directe per terminal i es mostren exemples.

9.2. Google Play

Qualsevol aplicació que compleixi els requisits existents es pot penjar a Google Play per fer-la accessible al públic des de qualsevol dispositiu Android. Per aconseguir això fa falta estar registrat al *Google Play Developer Console* i haver pagat 18€. Aquest pagament és únic i per sempre, al contrari que IOS, que requereix un pagament anual de 100€.

Un cop creat l'APK i registrat al Google Play es pot afegir una nova aplicació. Perquè es permeti publicar-la fan falta dos requeriments:

- Penjar un APK que compleixi els requisits establerts. Això significa que l'APK creat no ha de superar els 50MB, i en cas que ho fes, addicionalment es podria afegir contingut extra amb uns mètodes d'extensió. A més fa falta que l'APK estigui correctament signat amb un certificat virtual per identificar el creador de l'aplicació i que la seva versió no sigui igual o inferior que a les que ja s'han pujat anteriorment.

- Crear i documentar la pàgina d'informació de l'aplicació que es podrà veure a Google Play Apps. S'ha d'afegir una explicació de l'aplicació així com penjar imatges de com queda la *app* tant en mòbils com en tabletes. També s'ha d'especificar el tipus de públic al que està orientat i si es vol fer gratuïta o de pagament, entre d'altres petites qüestions.

Fig. **9.1.** Captura de pantalla del mòbil de la fitxa de Google Play de l'aplicació StarJet

Amb l'aplicació penjada al Google Play es poden realitzar diferents operacions interessants. Una d'elles es escollir com es vol penjar l'aplicació, si en mode alfa, beta, o promoció. Els modes alfa i beta serveixen per limitar l'accés dels usuaris a l'aplicació penjada si encara està en fase de proves, donant accés només a les persones que es vulgui per provar-la (testers). El mode promoció serveix per tenir l'aplicació completament oberta al públic. L'aplicació StarJet es troba actualment en fase Beta amb unes quantes persones afegides per fer proves fins que es decideixi com continuar-la i fins que el servidor estigui obert a temps complert i es permeti jugar correctament.

Una altre operació interessant a fer a Google Play és explorar les característiques que s'ofereixen en el panell de control del desenvolupador. Aquest panell permet veure les estadístiques del joc com el número d'instal·lacions, desinstal·lacions, errors apareguts, etc...

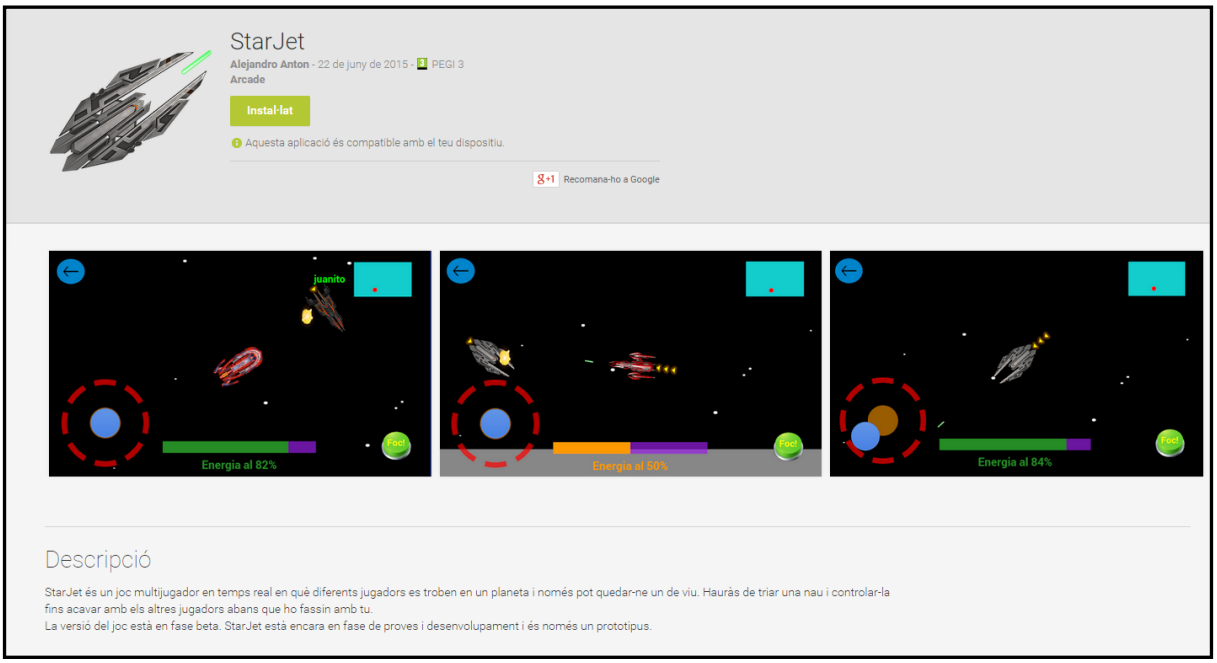

Fig. **9.2.** Captura de pantalla a l'ordinador de la fitxa de Google Play de l'aplicació StarJet

10. Mòduls complementaris

Hi ha tot un seguit de material que s'ha utilitzat en el desenvolupament que ha ajudat a que les coses fossin més fàcils i es fessin més ràpid. S'han creat alguns mòduls de Python per automatitzar certes qüestions, alguns d'ells han estat essencials i d'altres només s'han utilitzat per estalviar feina, com per exemple la automatització en la conversió dels fitxers ".ui" del Qt Designer a ".py", o la instal·lació de noves llibreries a Windows.

- **Convertidor d'interfícies**: Petita aplicació que permet automatitzar la conversió dels fitxers de QtDesigner a codi Python (fitxers ".ui" a ".py"). El que fa aquest programa no és més que executar la corresponent comanda de conversió de terminal de manera que tant l'entrada com la sortida queden guardades al directori on toca de forma instantània. S'ha utilitzat molt freqüentment per actualitzar la interfície gràfica del servidor multi-jugador. La comanda utilitzada seria: *pyuic4 fitxer_prova.ui –o fitxer_prova.py.*

- **Instal·lador de fitxers Wheel**: Petit mòdul que permet instal·lar automàticament les noves llibreries que es volen afegir a Python. Al estar treballant amb Windows les llibreries es poden instal·lar amb uns fitxers amb format *wheel* (".whl") i no amb fitxers executables (".exe") com fa poc temps. Es basa en el programa d'instal·lació de paquets *pip* que es crida des de la terminal. S'ha utilitzat per instal·lar les diferents llibreries utilitzades en el treball en diferents ordinadors. La comanda utilitzada seria *pip install llibreria_a_instalar.whl.*

- **Creador d'executables:** Petit mòdul que permet la creació de fitxers executables (".exe") d'aplicacions desenvolupades en Python. Aquest mòdul només crida de automàticament el fitxer *setup.py* que s'ha d'haver creat i que es basa en la llibreria de *py2exe*. Es va crear per aconseguir un arxiu executable de l'aplicació servidor per poder utilitzar-lo en altres ordinadors sense necessitat de tenir Python instal·lat.

Aquestes petites aplicacions creades es basen en el fet d'estalviar-se treballar amb la terminal de Windows, que es complica una mica més que amb Linux. Això ve a dir que amb Linux qualsevol programa instal·lat es pot cridar des de qualsevol directori, a Windows en canvi només es pot cridar des del directori en què es troba el fitxer. Aquesta qüestió es pot arreglar definint el programa que s'ha de cridar com una variable d'entorn de Windows, que significa que s'associa el fitxer amb el seu directori i es pot cridar des de qualsevol lloc. Si no es vol associar la variable d'entorn sempre es pot anar al directori de treball en qüestió.

```
import os
command = "pyuic4 fitxer_prova.ui -o fitxer_prova.py"
os.system(command)
```

Fig. **10.1.** Execució de la comanda de terminal del convertidor d'interfícies

11. Planificació

A continuació es mostra la planificació del projecte amb la distribució en el temps de les etapes i tasques realitzades. Aquesta planificació és l'estimació de la dedicació que han tingut realment totes les etapes del projecte.

S'estima que la inversió de temps en el projecte ha estat d'unes 380 hores, que s'ha allargat una mica més del que estava previst degut a l'abast que ha acabat tenint l'aplicació per mòbil així com la complexitat en el seu desenvolupament. El projecte es pot dividir en un seguit de tasques que es poden veure a continuació:

- **Recerca d'informació (5 hores*):*** Recerca de la tota la informació possible respecte les aplicacions per mòbil en general, i en concret en Python i amb *Kivy*.

- **Disseny general de l'aplicació (10 hores):** Plantejament i debat de l'estructura de l'aplicació en quan a funcionalitat i objectius basant-se en la recerca d'informació prèvia. En aquesta etapa s'ha pensat el funcionament de l'aplicació buscant quines eren les llibreries que podien utilitzar-se i que complien amb el que es volia fer.

- **Investigació i anàlisi de llibreries (30 hores):** Anàlisi exhaustiu de les llibreries que s'han utilitzat per preparar la programació de l'aplicació, fent extenses proves i simulacres d'elles així com de les seves interaccions.

- **Configuració de hardware (5 hores):** Anàlisi, instal·lació i configuració de tots els programes i material utilitzat en tots els dispositius, com el Oracle VM VirtualBox, configuració d'Eclipse i els programes citats anteriorment.

- **Desenvolupament del joc (130 hores):** Programació del joc en mode d'un sol jugador, programant l'estructura del joc amb tots els seus elements, classes i algorismes. També s'han programat totes les altres finestres com les d'ajustaments o crèdits així com també els missatges emergents com el d'inici de sessió

- **Desenvolupament del servidor (60 hores):** Programació del servidor amb la interfície gràfica i tots els seus elements. També inclou el disseny de la interfície.

- **Configuració multi-jugador (70 hores):** Adaptació del joc i del servidor al mode multi-jugador amb la interacció completa entre ells.

- **Posada apunt de l'aplicació (10 hores):** Adaptació de la programació del joc i del servidor per tenir certesa de que totes les variacions en el programa estan cobertes i no fallarà en cap punt, tant en mode d'un jugador com en multi-jugador.

- **Avaluació de software i proves (5 hores):** Exhaustives proves de validació i detecció d'errors de l'aplicació mòbil i del servidor recreant partides multi-jugador i intentant buscar situacions anòmales.

- **Conversió de Python a Android (10 hores):** Procés d'estudi i aplicació de la conversió dels fitxers de Python per crear l'APK.

- **Configuració de Google Play (5 hores):** Procés de configuració per disposar de l'aplicació mòbil a la plataforma Google Play per fer-la accessible al públic.

- **Documentació de la memòria (40 hores):** Recol·lecció de tota la feina feta per realitzar la memòria sencera i preparar la defensa oral.

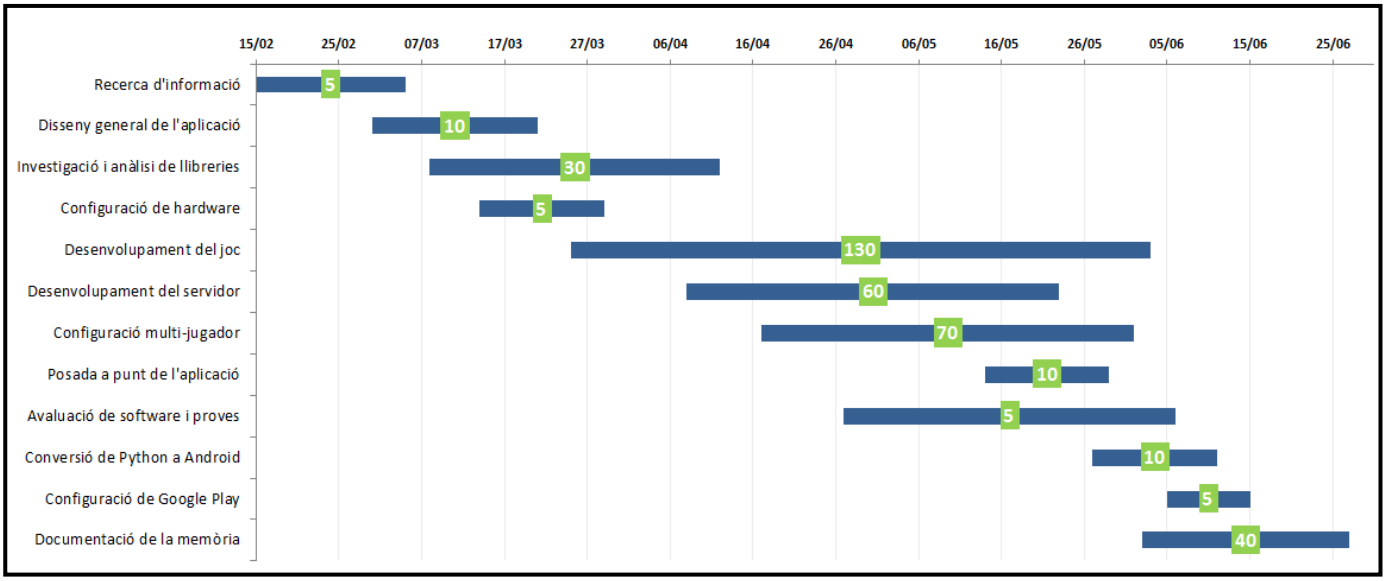

Fig. **11.1.** Diagrama de Gantt de les etapes del projecte amb la duració en hores a les etiquetes

12. Costos

En aquest apartat es mostren detalladament els costos de la realització del projecte que costen bàsicament de les hores de treball que cal pagar al dissenyador i al programador, així com altres costos menors en elements de hardware.

De les 380 hores invertides en el projecte la majoria es concentren en les tasques de programació, seguit del disseny i anàlisi de l'aplicació i de la documentació de la memòria.

En els costos s'han tingut en compte les 35 hores que consisteixen en la recerca d'informació i l'anàlisi de llibreries com a un cost d'investigació amb un preu menor, de 20 €/hora. Es considera que el preu a pagar a un enginyer industrial es de 35 €/hora i a un programador 30 €/hora.

Tots els elements de software que s'han utilitzat són gratuïts i no han generat costos.

Per afegir en el preu l'amortització dels elements de hardware utilitzats com l'ordinador i la tableta i el mòbil Samsung s'ha comptat amb que tenen una vida útil de 4 anys i que s'han fet servir durant aproximadament 4 mesos. Amb això s'obté una amortització a utilitzar d'aproximadament un 8.3%. El preu de l'ordinador és de 1300 € i el dels dispositius Samsung d'uns 700 € entre els dos.

Element	Professional	Duració (h)	Cost (€/h)	Cost (€)
Recerca i anàlisi de llibreries	Investigador	35	20	700
Disseny general de l'aplicació	Enginyer industrial	10	35	350
Configuració de hardware	Enginyer industrial	5	35	175
Tasques de programació i proves	Programador	275	30	8.250
Conversió a Android	Programador	10	30	300
Accés a Google Play	Enginyer industrial	5	35	175
El·laboració de la memòria	Enginyer industrial	40	35	1.400
Subscripció a Google Play				18
Amortització ordinador de taula i7				108
Amortització mòbil i tableta Samsung				58
Total		**380**		**11.535**

Fig. **12.1.** Taula de costos de tots els elements del projecte, tant tasques com material

Així doncs el cost total del projecte és de 11.535€.

13. Impacte ambiental

Al tractar-se d'un projecte de desenvolupament de software no es generen canvis susceptibles directes amb el medi ambient així com cap mena de contaminant. Tot i això, el projecte requereix l'ús constant dels mòbils i de l'ordinador que fa de servidor de l'aplicació. Aquests elements consumeixen electricitat i fan servir internet i per tant gasten energia i s'escalfen. Tot i tot el que s'ha esmentat l'impacte ambiental que genera aquest projecte es considera negligible.

Conclusions

S'ha aconseguit crear amb èxit l'aplicació amb l'arquitectura plantejada de bon inici al haver desenvolupat l'aplicació mòbil en temps real així com l'aplicació servidor. Ha estat una feina extensa però finalment s'ha aconseguit crear el prototipus del mini joc multi-jugador que es comunica amb el servidor i que aconsegueix accedir i emmagatzemar dades a la base MySQL. Amb això s'han aconseguit doncs, els objectius de la realització de les comunicacions web i de la base de dades que s'havien plantejat de bon inici.

Amb la realització d'aquest projecte crec que s'ha pogut demostrar la capacitat que té Python com a llenguatge de programació al haver treballat amb diversos camps no comuns d'ús amb aquest llenguatge i haver-los unit entre ells. S'ha pogut veure també que es pot disposar d'aplicacions fetes amb Python al web per fer-les accessibles al públic.

Totes les llibreries de Python que s'han utilitzat han estat molt interessants i molt útils i cal remarcar algunes coses d'elles. *Kivy* és una llibreria que tot i ser molt potent potser necessitaria ser una mica més clara i fàcil d'estructurar en alguns àmbits així com en la posterior conversió a Android, que és una mica llarga i complicada. La llibreria *PyQt4* ja s'havia fet servir anteriorment i tot i estar una mica antiquada en quan a documentació la segueixo recomanant per fer aplicacions amb interfície gràfica. La llibreria *SQLAlchemy* és per mi la millor eina per treballar amb bases de dades des de Python. Costa una mica entendre com funciona del tot però permet l'accés al món de les bases de dades d'una manera molt fàcil i la recomano profundament. Com a última llibreria a comentar, s'ha trobat *Twisted* una llibreria exemplar per treballar amb comunicació web. Tot i ser també una mica complicat d'entendre el seu funcionament, les portes que obre al poder fer aplicacions amb Python que requereixen connexió entre elles són moltíssimes.

En aquest previ anàlisi de les llibreries utilitzades no s'ha mencionat si recomano o no la llibreria *Kivy* com a eina per desenvolupar aplicacions per mòbil amb Python. El que jo opino és que si realment es vol fer servir aquest llenguatge per fer una aplicació, aleshores endavant, ja que s'obtindran grans resultats, però si s'ha de dubtar hi han eines més senzilles com Java o UNITY. Per fer jocs sempre hi hauran millors opcions que *Kivy.* Si es volen fer amb *Kivy* aplicacions mínimament complicades que requereixen per exemple comunicació web d'una manera simple, també recomano el seu ús però amb moderació, ja que requerirà molt esforç. El que no recomano en cap cas és la realització d'aplicacions per mòbil amb *Kivy* que requereixin molts càlculs en els dispositius perquè els costa bastant processar, com els jocs que requereixen actualitzacions per segon com StarJet. Tampoc recomano el desenvolupament de jocs online que funcionin en temps real, ja que és molt important la precisió a la què arriben les dades i és un món molt complicat i problemàtic.

Futures accions

De cara a la feina que es podria fer un cop acabat el treball si es volgués continuar amb l'aplicació, hi ha coses que s'ha pensat que es podrien fer i millorar. Algunes d'elles són:

- Estudiar la possibilitat de prescindir de la feina d'intermediari que realitza el servidor un cop començada una partida, és a dir, que la comunicació es dugui a terme directament de mòbil a mòbil i només s'accedeixi al servidor per agafar i emmagatzemar les dades abans i després de jugar.

- Fer un petit estudi sobre el protocol d'ús de les connexions a l'aplicació, si seria millor utilitzar **UDP** o deixar-lo a **TCP** com està actualment.

- El joc hauria de ser revisat en qüestions de rendiment de *frames* (FPS) ja que a vegades li costa treballar i depèn lleugerament de la qualitat del dispositiu.

- Acabar de perfilar l'estructura completa de l'actualització de dades del joc multi-jugador. Això inclou els casos en què la connexió entre dos jugadors s'alenteix o es tanca, predient la posició que tindria l'altre jugador mentre no arribin les dades necessàries de refrescament d'informació.

- Si el joc es portés a terme a nivell empresarial i tingués èxit, s'estudiaria la possibilitat d'introduir publicitat per obtenir ingressos. Això no seria fàcil ja que el codi i configuració de la publicitat va inclosa en el codi de l'aplicació, i la majoria de plataformes de publicitat per a aplicacions ofereixen els seus serveis en el llenguatge d'Android, que és Java.

- Si el joc disposés de molts jugadors el que s'hauria de fer és modificar lleugerament l'estructura del servidor. El servidor actualitza en temps real els esdeveniments que hi tenen lloc i modifica tots els elements de la interfície gràfica, i això amb molts jugadors consumiria masses recursos. S'hauria d'eliminar la interfície o modificar-la de manera que només mostrés l'estat del joc refrescant-ho de manera puntual.

- Alguns algorismes del joc haurien de ser revistats com el de col·lisions i reflexió amb les parets i s'estudiarien possibles millores per afegir al joc. Algunes idees serien afegir una llista d'amics, poder modificar el nom d'usuari i la contrasenya, poder jugar com un convidat, i afegir traduccions al castellà i a l'anglès.

Si la decisió final fos de no continuar amb aquesta aplicació amb Python, trobaria molt interesant començar a introduir-me en la creació d'aplicacions amb UNITY.

Agraïments

En primer lloc m'agradaria agrair l'ajuda, dedicació i interès que m'ha proporcionat el meu director del treball i professor Lluís Solano al llarg de tot el projecte. Vull agrair-li que hagi estat sempre disponible per comentar qüestions del treball així com per guiar-me durant l'elaboració de l'aplicació i de la memòria.

M'agradaria també agrair l'ajuda a meu amic Joan Heredia i a la meva parella Queralt Borrell per aportar idees i consells per al desenvolupament del joc així com per ajudar a provar l'aplicació multi-jugador per trobar errors i qüestions a millorar.

Bibliografia

Referències bibliogràfiques

[1] BLOG.CODEEVAL. Most popular coding languages of 2015.
[http://blog.codeeval.com/codeevalblog/2015#.VZlI__ntlBc=/, 25 de Juny de 2015]

Bibliografia complementària

[1] KIVY. Home. [http://kivy.org/#home/, 25 de Febrer de 2015]

[2] LFD. Unofficial Windows binaries for Python extension packages.
[http://www.lfd.uci.edu/~gohlke/pythonlibs/, 5 de Març de 2015]

[3] PYQT.SOURCEFORGE. PyQt Class Reference .
[http://pyqt.sourceforge.net/Docs/PyQt4/classes.html, 20 de Març de 2015]

[4] ICONFINDER. Free icons!
[https://www.iconfinder.com/free_icons, 3 de Maig de 2015]

[5] TWISTED MATRIX. Twisted Examples.
[https://twistedmatrix.com/documents/current/core/examples/, 10 d'Abril de 2015]

[6] GITHUB. Python for Android Usage.
[https://github.com/kivy/python-for-android/blob/master/docs/source/usage.rst, 28 de Maig de 2015]

[7] DEVELOPER.ANDROID. Signing your Applications.
[http://developer.android.com/tools/publishing/app-signing.html, 2 de Juny de 2015]

[8] DOCS.SQLALCHEMY. Engine configuration.
[http://docs.sqlalchemy.org/en/rel_0_8/core/engines.html, 21 d'Abril de 2015]

[9] DOCS.SQLALCHEMY. Query API.
[http://docs.sqlalchemy.org/en/latest/orm/query.html, 23 d'Abril de 2015]

[10] AVATAR. Colour names.
[http://www.avatar.se/molscript/doc/colour_names.html, 29 de Març de 2015]

[11] FREESOUND. Home.
[http://www.freesound.org/, 14 de Maig de 2015]

[12] GOOGLE PLAY. Google Play Developer Console.
[https://play.google.com/apps/publish/?dev_acc=*****#AppListPlace, 5 de Juny de 2015]

[13] MILLIONTHVECTOR. Free sprites.
[http://millionthvector.blogspot.com.es/p/free-sprites_12.html, 29 de Març de 2015]

[14] KIVYSPACEGAME. Tutorial.
[https://kivyspacegame.wordpress.com/category/kivy/, 8 de Maig de 2015]

[15] GAFFERONGAMES. Game networking.
[http://gafferongames.com/networking-for-game-programmers/, 25 d'Abril de 2015]

[16] OPENGAMEART. Textures. [http://opengameart.org/textures/, 15 de Maig de 2015]

[17] GAMEDEVELOPMENT. Building a multiplayer networked game.
[http://gamedevelopment.tutsplus.com/tutorials/building-a-peer-to-peer-multiplayer-networked-game--gamedev-10074, 2 d'Abril de 2015]

[18] WEBZO. Space Shooter pause and sound.
[http://www.webzo.org/tutorials/flash/space-shooter-pause-game-sound.php, 23 de Març de 2015]

[19] STACKOVERFLOW. Movement algorithm in client-server multiplayer game.
[http://stackoverflow.com/questions/1065758/movement-algorithm-in-client-server-multiplayer-mmo-games, 17 d'Abril de 2015]

[20] PETERCOLLINGRIDGE. Pygame physics simulation tutorial.
[http://www.petercollingridge.co.uk/book/export/html/6, 9 de Març de 2015]

[21] DEVELOPER.ANDROID. Manifest permissions for Applications.
[http://developer.android.com/reference/android/Manifest.permission.html, 26 de Maig de 2015]

[22] GITHUB. Kivy recipes to add to a distribution.
[https://github.com/kivy/python-for-android/tree/master/recipes, 21 de Maig de 2015]

[23] STACKOVERFLOW. Request from a data base.
[http://stackoverflow.com/questions/14340105/should-a-multiplayer-game-always-request-data-from-a-database-on-each-client-req, 12 de Març de 2015]

[24] FLASK.POCOO. Flask API [http://flask.pocoo.org/docs/0.10/api/, 9 de Març de 2015]

[25] GITHUB. Using the QtReactor.
[https://github.com/ghtdak/qtreactor, 12 d'Abril de 2015]

[26] DOCS-PYTHON. Cryptography.
[http://docs.python-guide.org/en/latest/scenarios/crypto/, 16 de Maig de 2015]

[27] KIVY. Integrating with other framworks – Twisted.
[http://kivy.org/docs/guide/other-frameworks.html, 10 d'Abril de 2015]